TP N°1 : Classes et Objets

Exercice 1 :

1. Définissez une classe appelée CompteBancaire.
2. La classe CompteBancaire devrait avoir les attributs suivants :
   - titulaire : le nom du titulaire du compte.
   - solde : le solde actuel du compte.
3. Ajoutez une méthode afficher_infos qui affiche le nom du titulaire et le solde du compte.
4. Ajoutez une méthode deposer qui prend un montant en paramètre et l'ajoute au solde du compte.
5. Ajoutez une méthode retirer qui prend un montant en paramètre et le soustrait du solde du compte, à condition que le solde reste positif.
6. Créez deux instances de la classe CompteBancaire avec des titulaires et des soldes différents.
7. Affichez les informations de chaque compte.
8. Effectuez des opérations de dépôt et de retrait sur l'un des comptes et affichez à nouveau les informations pour vérifier les changements.

Exercice 2 :

1. Définir une classe Client avec les attributs suivants : CIN, Nom, Prénom, Tél.
2. Définir les méthodes d'accès aux différents attributs de la classe.
3. Définir un constructeur permettant d'initialiser tous les attributs.
4. Définir un constructeur permettant d'initialiser le CIN, le nom et le prénom.
5. Définir la méthode Afficher ( ) permettant d'afficher les informations du Client en cours.
6. Créer Une classe Compte caractérisée par son solde et un code qui est incrémenté lors de sa création ainsi que son propriétaire qui représente un client.
7. Définir les méthodes d'accès aux différents attributs de la classe (le numéro de compte et le solde sont en lecture seule)
8. Définir un constructeur permettant de créer un compte en indiquant son propriétaire (client).
9. Ajouter à la classe Compte les méthodes suivantes :
    a. Une méthode permettant de Crediter() le compte, prenant une somme en paramètre.
    b. Une méthode permettant de Debiter() le compte, prenant une somme en paramètre
    c. Une méthode qui permet d'afficher le résumé d'un compte.
    d. Une méthode qui permet d'afficher le nombre des comptes crées.
10. Créer un programme de test pour la classe Compte.

Exercice 3 :

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 4 :

Ecrivez une classe abstraite Employé avec les attributs suivants :

- Matricule
- Nom
- Prénom
- Date de naissance

La classe Employé doit disposer des méthodes suivantes :

- Un constructeur d'initialisation
- Des getters et setters pour les différents attributs
- La méthode __str__()
- Une méthode abstraite GetSalaire().

Un **ouvrier** est **un** employé qui se caractérise en plus par sa date d'entrée à la société.
Tous les ouvriers ont une valeur commune appelée SMIG=3000 DH
L'ouvrier a un salaire mensuel qui est :
Salaire = SMIG + (Ancienneté en année) *100.
De plus, le salaire ne doit pas dépasser SMIG*2.

Un **cadre** est un employé qui se caractérise par un indice.
Le cadre a un salaire qui dépend de son indice :
1 : salaire annuel brut 130000 DH
2 : salaire annuel brut 150000 DH
3 : salaire annuel brut 170000 DH
4 : salaire annuel brut 200000 DH

Un **patron** est un employé qui se caractérise par un chiffre d'affaires et un pourcentage.
Le chiffre d'affaires est commun entre les patrons.
Le patron a un salaire annuel qui est égal à x% du chiffre d'affaires :
Salaire = CA*pourcentage/100

Travail à faire :

- Créer la classe abstraite Employé.
- Créer la classe Ouvrier, la classe Cadre et la classe Patron qui héritent de la classe Personne.
- Créer un programme de test pour tester les différentes classes.

Exercice 5 :

Créer une classe **Etudiant** caractérisée par nom, âge et moyenne.

- L'âge doit être entre 18 et 26 sinon une exception est levée (elle affiche le message "L'âge doit être entre 18 et 26") est générée.
- La note doit être entre 0 et 20 sinon une exception est levée (elle affiche le message "La note doit être entre 0 et 20").

Définir le constructeur d'initialisation de la classe, les getters/setters et la méthode __Str__()
Créer un programme qui permet de tester la classe.

Exercice 6 :

A. Créer la classe stagiaire caractérisée par :

1. Les attributs : CIN, Nom, Prénom, Filière, Note
2. Les propriétés.
3. Un constructeur d'initialisation
4. La méthode __str__()

B. Créer un programme qui manipule une **liste de stagiaires** par un ensemble d'opérations sous forme d'un menu comme suit :

1. Afficher tous les stagiaires.
2. Afficher les notes de tous les stagiaires.
3. Afficher les stagiaires ayant une note supérieure ou égale à une note donné.
4. Ajouter un stagiaire dont les informations sont entrées par l'utilisateur. (CIN Unique)
5. Rechercher un stagiaire par CIN donné.
6. Rechercher les stagiaires d'une filière donnée.
7. Supprimer un stagiaire dont le CIN est entré par l'utilisateur.
8. Quitter

Exercice 7 :

A. Créer la classe **stagiaire** caractérisée par :

1. Les attributs : CNE, Nom, Prénom
2. Les propriétés.
3. Un constructeur d'initialisation
4. La méthode __str__()

B. Créez une classe **Groupe** caractérisée par :

1. Les attributs : le nom du groupe, la filière, et une liste de stagiaires.
2. Les propriétés.
3. Un constructeur d'initialisation
4. La méthode __str__()
5. Une méthode qui permet de rechercher un stagiaire par son CNE dans le groupe.
6. Une méthode qui vérifie si un stagiaire spécifique (CNE) existe dans le groupe.
7. Une méthode qui vérifie si un stagiaire spécifique (Nom et prénom) existe dans le groupe.
8. Une méthode qui permet d'ajouter un stagiaire au groupe (vérifiez que le CNE est unique et qu'il n'y a pas déjà 25 stagiaires dans le groupe).
9. Une méthode qui permet d'afficher la liste des stagiaires dans le groupe.
10. Une méthode qui permet de retirer un stagiaire du groupe à partir de son CNE.

C. Testez votre classe en créant un groupe, en ajoutant quelques stagiaires, en affichant la liste des stagiaires, en retirant un stagiaire, et en vérifiant que la limite de 24 stagiaires est respectée, etc.

Exercice 8 :

Vous êtes responsable de la gestion des stocks d'une boutique en ligne. Chaque produit en vente est représenté par un dictionnaire contenant les informations suivantes :

- Nom du produit
- Prix unitaire
- Quantité en stock

Créez une liste de dictionnaires pour stocker les informations sur plusieurs produits.

Écrivez un programme sous forme de menu pour effectuer les opérations suivantes :

- Ajouter un nouveau produit à la liste avec son nom, son prix unitaire et sa quantité en stock.
- Rechercher un produit en utilisant son nom, et afficher toutes ses informations.
- Afficher la liste de tous les produits avec leurs informations.
- Mettre à jour les informations d'un produit existant en utilisant son nom.
- Supprimer un produit de la liste en utilisant son nom.