Module : POO Python (M103)
TP N° 4

Année de Formation 2023/2024

Filière : Développement digital Groupe : DEV 101 - DEV 102 Niveau : 1ère année

Exercice 1 :

1. Créez une classe `Personne` avec les attributs privés (en utilisant `_`) suivants :

- `**_nom**` (nom de la personne)

- `**_age**` (âge de la personne)

2. Ajoutez un constructeur `__init__` à la classe `Personne` qui prend en compte les deux attributs privés, `_nom` et `_age`, et initialise ces attributs.

3. Ajoutez une méthode publique `afficher_informations` à la classe `Personne` qui affiche le nom et l'âge de la personne.

4. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_nom` et `_age` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

Partie 2 : Classe Employé

1. Créez une classe `Employe` qui hérite de la classe `Personne`.

2. Ajoutez les attributs privés suivants à la classe `Employe` :

- `**_poste**` (poste de l'employé)

- `**_salaire**` (salaire de l'employé)

3. Ajoutez un constructeur `__init__` à la classe `Employe` qui prend en compte les attributs `_nom`, `_age`, `_poste`, et `_salaire`, et initialise ces attributs. Assurez-vous d'appeler le constructeur de la classe de base (`Personne`) pour initialiser les attributs `_nom` et `_age`.

4. Ajoutez une méthode publique `afficher_informations` à la classe `Employe` qui affiche le nom, l'âge, le poste, et le salaire de l'employé. Assurez-vous d'appeler la méthode `afficher_informations` de la classe de base (`Personne`) pour afficher le nom et l'âge.

5. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_poste` et `_salaire` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

Partie 3 : Test de l'héritage avec encapsulation

1. Créez une instance de la classe `Personne` et une instance de la classe `Employe`.

2. Utilisez les méthodes d'accès pour modifier les attributs des instances de la classe `Employe` et affichez les informations de l'employé après les modifications.

Exercice 2 :

Partie 1 : Classe Personne

1. Créez une classe `Personne` avec les attributs privés suivants (en utilisant `_`) :

- `**_nom**` (nom de la personne)
- `**_age**` (âge de la personne)
- `**_adresse**` (adresse de la personne)

2. Ajoutez un constructeur `__init__` à la classe `Personne` qui prend en compte les trois attributs privés, `_nom`, `_age`, et `_adresse`, et initialise ces attributs.

3. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_nom`, `_age`, et `_adresse` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

Partie 2 : Classe Etudiant

1. Créez une classe `Etudiant` qui hérite de la classe `Personne`.

2. Ajoutez les attributs privés suivants à la classe `Etudiant` :

- `_matricule` (numéro de matricule de l'étudiant)

- `_classe` (nom de la classe à laquelle l'étudiant est inscrit)

3. Ajoutez un constructeur `__init__` à la classe `Etudiant` qui prend en compte les attributs `_nom`, `_age`, `_adresse`, `_matricule`, et `_classe`, et initialise ces attributs. Assurez-vous d'appeler le constructeur de la classe de base (`Personne`) pour initialiser les attributs `_nom`, `_age`, et `_adresse`.

4. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_matricule` et `_classe` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

Partie 3 : Test de l'héritage avec encapsulation

1. Créez une instance de la classe `Etudiant` pour représenter un étudiant inscrit dans une classe.

2. Utilisez les méthodes d'accès pour afficher les informations de l'étudiant, y compris le nom, l'âge, l'adresse, le matricule et la classe.

3. Modifiez l'adresse de l'étudiant en utilisant la méthode de modification appropriée et affichez les informations mises à jour de l'étudiant.

Exercice 3 :

Partie 1 : Classe Employe

1. Créez une classe `Employe` avec les attributs privés suivants (en utilisant `_`) :

- `**_nom**` (nom de l'employé)

- `**_age**` (âge de l'employé)

- `**_salaire**` (salaire de l'employé)

2. Ajoutez un constructeur `__init__` à la classe `Employe` qui prend en compte les trois attributs privés, `_nom`, `_age`, et `_salaire`, et initialise ces attributs.

3. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_nom`, `_age`, et `_salaire` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

Partie 2 : Classe Manager

1. Créez une classe `Manager` qui hérite de la classe `Employe`.

2. Ajoutez les attributs privés suivants à la classe `Manager` :

- `_departement` (département géré par le manager)

- `_subordonnes` (une liste des employés sous la supervision du manager)

3. Ajoutez un constructeur `__init__` à la classe `Manager` qui prend en compte les attributs `_nom`, `_age`, `_salaire`, `_departement`, et `_subordonnes`, et initialise ces attributs. Assurez-vous d'appeler le constructeur de la classe de base (`Employe`) pour initialiser les attributs `_nom`, `_age`, et `_salaire`.

4. Ajoutez des méthodes d'accès (getters et setters) pour les attributs `_departement` et `_subordonnes` pour permettre la lecture et la modification de ces attributs depuis l'extérieur de la classe.

5. Ajoutez une méthode `ajouter_subordonne` qui prend en compte un objet `Employe` et ajoute cet employé à la liste `_subordonnes` du manager.

Partie 3 : Test de l'héritage avec encapsulation

1. Créez une instance de la classe `Employe` pour représenter un employé de base.

2. Créez une instance de la classe `Manager` pour représenter un manager supervisant des employés.

3. Utilisez les méthodes d'accès pour afficher les informations de l'employé et du manager, y compris le nom, l'âge, le salaire, le département, et la liste des subordonnés.

4. Utilisez la méthode `ajouter_subordonne` pour ajouter un ou plusieurs employés à la liste des subordonnés du manager, puis affichez la liste mise à jour des subordonnés du manager.