ITI 1520 - Devoir 5

Date de remise: **le 4 decembre, avant 22h00**

Instructions

Vous devez faire ce travail en groupes de deux étudiant(e)s. Incluez des commentaires avec les deux noms et les deux numéros d'étudiant et affichez-les au début de chaque programme. Un(e) des deux membres du groupe seulement doit soumettre le devoir dans le campus virtuel, mais il est important d'indiquer le nom du (de la) deuxième étudiant(e) ainsi que son numéro d'étudiant dans les commentaires et les noms des fichier .py pour qu'il (elle) reçoive lui (elle) aussi la note.

Répondez à la question 1 dans le fichier D5Q1.py. Utilisez D5Q1.py comme point de départ, seulement le programme principal est complété. **Complétez toutes les fonctions et les classes qui contiennent le commentaire '# à compléter'**),
Répondez à la question 2 dans le fichier D5Q2.py.
Mettez tous les fichiers dans un répertoire compressé **d5_numEtudiant1_numEtudiant2.zip** pour soumission dans le campus virtuel.

Ajouter des **commentaires** dans chaque programme pour expliquer le but du programme, la fonctionnalité de chaque fonction et le type des paramètres et du résultat, et des commentaires pour expliquer votre programme. Des points vous seront enlevés si les commentaires sont déficients.

Question 1 (12 points) Gestion de campus

Dans ce travail, vous devez implémenter les classes: *Personne*, *Etudiant*, *Employe* et *Gestion.*
On fournit le diagramme UML contenant la description de toutes les classes.
Le programme principal est déjà implémenté dans le fichier *d5q1.py*. Vous devez donc compléter toutes les classes.

Classe Personne
La classe *Personne* comprendra les attributs suivants :

- *nom* est une chaine de caractères.
- *prénom* est une chaine de caractères.
- *identifiant* est un nombre entier.

Cette classe dispose les méthodes suivantes :

- La méthode `__init__()` doit initialiser le nom d'un objet de type classe Personne, son prénom et son identifiant. Cette fonction prend en entrée un nom, un prénom de type chaine de caractères et un identifiant de type entier.
- La méthode `_repr_()` doit retourner une chaine de caractères qui contient tous les informations d'une personne.
- La méthode `_eq_()` doit vérifier si deux personnes sont identiques (le même nom, le même prénom et le même identifiant). Cette fonction elle prend en entrée un objet de type *Personne* et retourne *True* si les deux personnes sont identiques et *False* sinon.

Classe Employe

La classe *Employe* dérive de la classe *Personne*. Cette classe comprendra les attributs suivants :

- *tauxHoraire* est un nombre réel.

Cette classe dispose les méthodes suivantes :

- La méthode `__init__()` doit initialiser un objet de type *Employe*. Cette fonction prend en entrée un nom, un prénom de type chaine de caractères, un identifiant de type entier et un taux de type réel.
- La méthode `_repr_()` doit retourner une chaine de caractères qui contient tous les informations d'un employé.
- La méthode `calculerSalaire()` doit retourner le salaire d'un employé. Elle prend en paramètre le nombre des heures (un nombre entier). Le salaire égale le nombre des heures multiplié par le taux horaire.

Classe Etudiant

La classe *Etudiant* dérive de la classe *Personne*. Cette classe comprendra les attributs suivants :

- *solde* est un nombre réel (présente le solde à payer pour les frais de scolarité).
- *cours* est une liste de cours (une liste de chaine de caractères).

Cette classe dispose les méthodes suivantes :

- La méthode `__init__()` doit initialiser un objet de type *Etudiant*. Cette fonction prend en entrée un nom, un prénom de type chaine de caractères, un identifiant de type entier, un solde de type réel et une liste de cours. La liste de cours doit être initialisée à une liste de chaine de caractères vide.
- La méthode `_repr_()` doit retourner une chaine de caractères qui contient toutes les informations d'un étudiant. Pour la liste de cours, ajouter seulement le nombre de cours dans la chaine de caractères.
- La méthode `ajouterCours()` doit ajouter un cours dans la liste de cours. Elle prend en paramètre le nom de cours (une chaine de caractères). Un étudiant peut ajouter un cours seulement si il n'a aucun solde à payer (la valeur de solde est égale à zéro). Cette fonction retourne *True* si le cours a été ajouté dans la liste de cours et *False* sinon.

Classe Gestion

La classe *Gestion* permet de gérer la liste des étudiants et des employés. Cette classe comprendra deux atributs (statiques) :

- Une liste des étudiants *listEtudiant*.
- Une liste des employés *listEmploye*.

Cette classe ne comprend pas aucuns attributs.

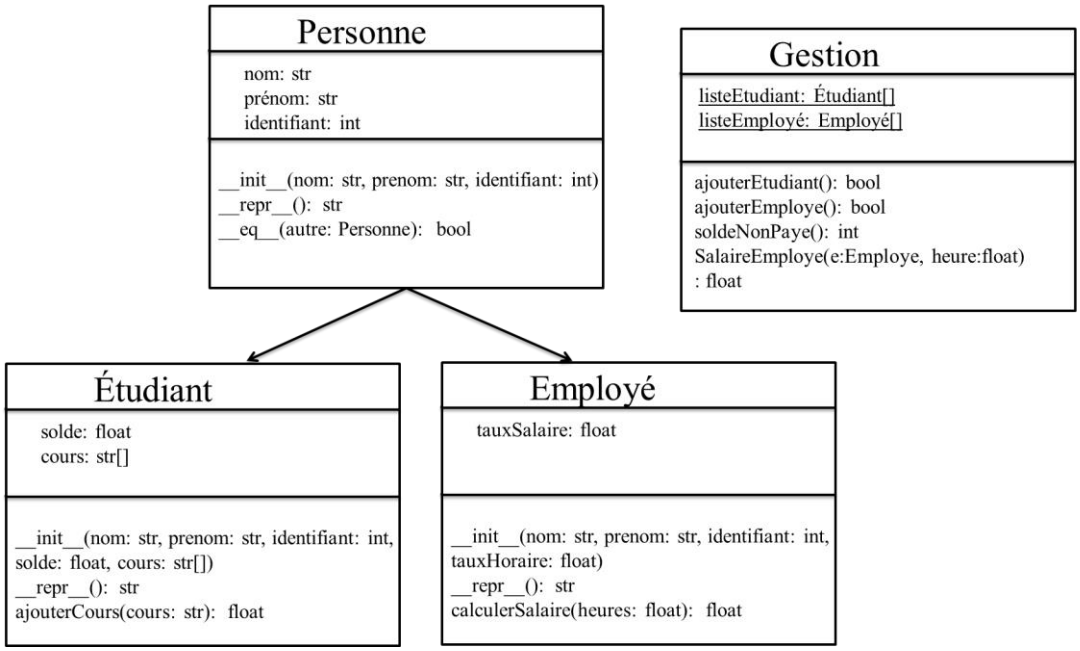
Cette classe dispose les méthodes suivantes :

- La méthode `ajouterEtudiant()`. Cette fonction ne prend aucun paramètre en entrée. Elle permet de demander à l'utilisateur les informations pour l'étudiant à ajouter. Par la suite, elle doit demander à l'utilisateur d'entrer un choix : si l'utilisateur veut ajouter un cours (le choix de l'utilisateur doit être oui ou non, sinon afficher un message approprié). Avant d'ajouter l'étudiant, il faut vérifier qu'il n'existe pas dans la liste. Cette fonction elle doit retourner *True* si l'étudiant a été ajouté et *False* sinon.
- La méthode `ajouterEmploye()`. Cette fonction ne prend aucun paramètre en entrée. Elle permet de demander à l'utilisateur les informations pour l'employé à ajouter. Avant d'ajouter

l'employé, il faut vérifier qu'il n'existe pas dans la liste. Cette fonction elle doit retourner *True* si l'employé a été ajouté et *False* sinon.

- La méthode `soldeNonPaye()`. Cette fonction ne prend aucun paramètre en entrée. Elle permet de calculer et retourner le nombre des étudiants qui ont un solde non payé.
- La méthode `salaireEmploye()`. Cette fonction prend en entrée un employé et le nombre des heures travaillées. Elle permet de calculer et retourner le salaire d'un employé. Si l'employé n'existe pas dans la liste, le salaire est égal à zéro.

Le diagramme UML suivant représente la description des classes.



Question 2 (8 points)

a. Écrivez une fonction Python **récursive** appelée `triangle` qui prendra comme paramètre un entier non négatif et qui générera un dessin composé d'étoiles tel qu'affiché ci-dessous. Vous pouvez utiliser une boucle pour générer une ligne d'étoiles, mais pas le dessin en entier.

Exemple :

```
>>> triangle(5)
*
**
***
****
*****
```

b. Écrivez une fonction Python **récursive** appelée `prodListePos_rec` qui prendra comme paramètre une liste et comme deuxième paramètre le nombre des éléments de la liste, et qui retournera le produit des éléments positifs (> 0).

Exemple :

```
>>> l = [1,-2,5,0,6,-5]
>>> prodListePos_rec(l, len(l))
30
```

Note: Si les fonctions pour Q2 a, b **ne sont pas récursives**, le nombre de points reçus sera **zéro**. (Vous pouvez tester des fonctions équivalentes itératives pour vous-même.)