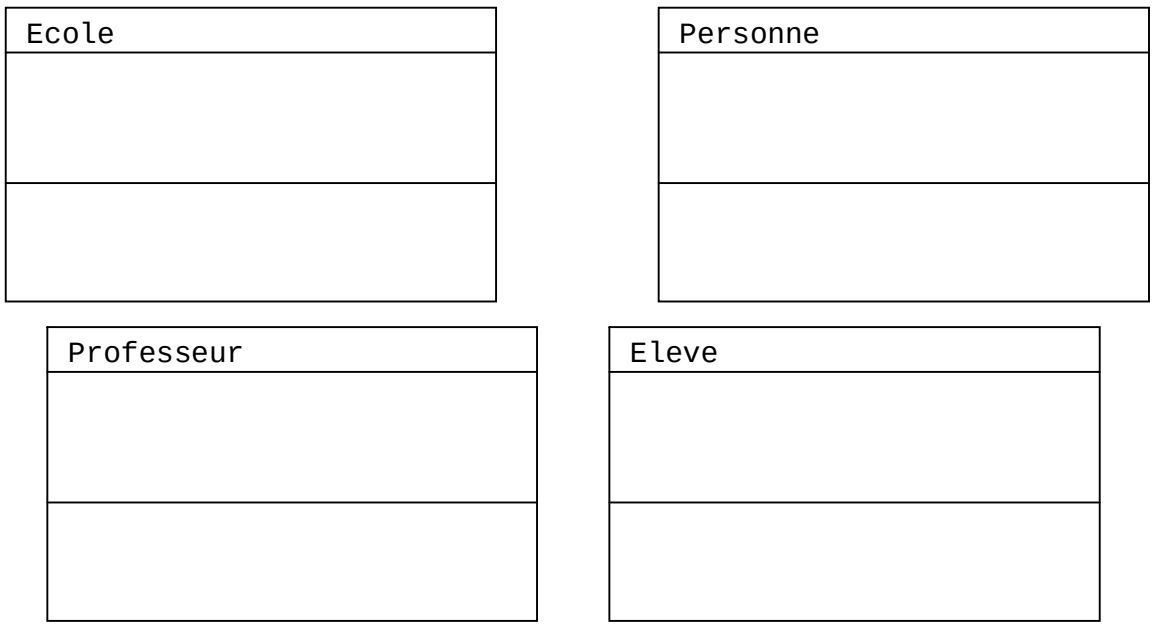
TP n10 : Révisions

Vous allez devoir créer une application permettant de gérer le fonctionnement d'une école. Une école est composée d'élèves et de professeurs. Nous allons utiliser les différents concepts vus en cours pour réaliser cette application. A chaque fin de paragraphe, appelez-moi pour contrôler. Par ailleurs, une fois fini, zippez votre projet et utilisez cette adresse pour l'envoyer : http://tibo.lelore.free.fr/IRIS/1Annee

Relation entre les classes

Complétez le diagramme UML suivant, en mettant des flèches quand il y a un héritage, et un lien simple quand une classe contient une autre classe :



Attributs et Méthodes

Complétez le diagramme précédent pour faire apparaitre les informations suivantes (en précisant le type) :

- Une **école** possède un *nom*, une *adresse* et un *budget* (un nombre qui désigne l'argent disponible)
- Les **Professeur** et les **Eleve** ont un *nom*, un *prénom* et un age.
- Un **Professeur** a un *salaire*
- Un **Eleve** a un tableau de *moyenne* (un tableau de 5 entiers qui équivalent à 5 matières…)

En plus de ces attributs, faites apparaitre ces fonctions :

- **`void afficher()`** : permet d'afficher les informations de la classe (à faire pour toutes les classes)
- **`void vieillir()`** : permet de rajouter 1 à l'attribut age.
- **`int calculCout()`** : calcule le coût de revient d'une professeur sur 1 an
- **`int moyenneGen()`** : calcule la moyenne générale d'un élève (moyenne des 5 matières)

Constructeur et Destructeurs

Enfin, avant que tout soit prêt à être implémenter, réfléchissez aux constructeurs et destructeurs :

- Combien de paramètres faut-il au constructeur de **Professeur** ?
- Même question pour le constructeur de **Eleve** ?
- Même question pour le constructeur de **Personne** ?
- Dans quel ordre sont appelés les constructeurs dans un héritage :
    - Le constructeur de **Personne** puis celui de **Eleve**
    - Le constructeur de **Eleve** puis celui de Personne
    - Uniquement celui de **Eleve**
    - Uniquement celui de **Personne**
- Dans quel ordre sont appelés les destructeurs dans un héritage :
    - Le destructeur de **Personne** puis celui de **Eleve**
    - Le destructeur de **Eleve** puis celui de Personne
    - Uniquement celui de **Eleve**
    - Uniquement celui de **Personne**

Question intermédiaire

On souhaite pouvoir compter le nombre total d'objets de type **Professeur** et le nombre total d'objets de type **Eleve**. Donnez une solution à ce problème en complétant le diagramme UML précédent.

Implémentation

Commencez à faire le code C++ de la classe **Personne**. Une fois cette classe finie, testez là avec une fonction main où vous créerez une personne, vous l'afficherez, vous la ferez vieillir puis vous la réafficherez. Bref, testez toutes les fonctions que vous avez créées.

Une fois les différentes fonctions testées, et validées, implémentez la classe **Eleve**. De même, testez toutes les fonctions que la classe contient (y compris les fonctions héritées) dans le main.

Ensuite, faites la classe **Professeur**, et testez de la même manière, testez le comportement de la classe.

Ecole

Maintenant que vous avez créé les différents objets, occupons nous de la classe **Ecole**. L'idée est de dire qu'il y a toujours 2 professeurs dans une école et 5 élèves. Ajoutez donc dans la classe **Ecole** deux tableaux, l'un de type **Professeur**, l'autre de type **Eleve**. Puisque vous utilisez des tableaux, il faut que vous ajoutiez un constructeur simple qui ne prend pas de paramètres (qui met tout à 0) aux deux classes… Ensuite, faites le constructeur de la classe **Ecole** qui prend un nom, une adresse et un budget en paramètre (voir le diagramme UML). Dans la fonction **`afficher`**, il faut aussi que vous affichiez les infos des **Eleve** et des **Professeur**, ainsi que leur nombre (en utilisant la variable **static**).

Faites un main où vous créerez une **Ecole**, et testez la fonction **`afficher()`**… Expliquez pourquoi il n'y a ni le nom des professeurs, ni le nom des élèves… Me proposer une solution.