Introduction aux TDs en Java

Bonne pratique : toujours refléchir en avance à son code, sur un bout de papier, avant de commencer à l'implémenter ! Ceci permettra à vos programmes de gagner de la visibilité et la clarté.

Nos tâches principales pendant les TDs :

- Au début : des petits exercices pour s'habituer à la programmation Java ;
- Comprendre ce qu'un bout de code va donner comme résultat ;
- Essayer d'anticiper, de détecter et de corriger des erreurs dans le code ;
- Après : réfléchir à comment concevoir et designer son code ;
- Vous allez également avoir le module COO, qui va vous aider avec la conception des objets, etc.

Nous n'allons pas utiliser nos ordinateurs en TD, le but étant justement de réussir anticiper le comportement d'une compilation ou d'une exécution Java.

Aujourd'hui : TD 1 Les classes et comment les utiliser

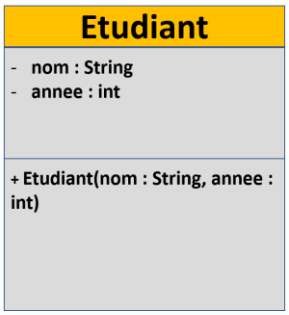
Exercice I

On commence par une classe `Etudiant`. Dans celle-ci nous allons avoir les prochains attributs :

- Un attribut `nom` de type `String` ;
- Un attribut `annee` de type entier ;

De plus, nous aurons mettre en place un constructeur avec la signature `Etudiant(String nom, int annee)`

Cette classe peut également être représentée par un *diagramme de classe* comme ci-dessous :



Nous allons voir les diagrammes de classe plus tard dans ce cours. Pour l'instant, prenons-les comme une illustration sommaire de la classe. Il faut savoir qu'un petit « - » devant un attribut ou une méthode veut dire que celui/celle-ci est privé(e), tandis qu'un petit « + » veut dire que l'attribut ou la méthode est publique.

Nous allons premièrement détailler cette classe.

1. Petit rappel :
    - Qu'est-ce qu'un attribut ?
    - Qu'est-ce qu'un constructeur ?

2. Le premier constructeur initialisera les attributs de la classe par les valeurs données en entrée. Ecrivez ce constructeur.

3. Etudiez le code suivant :

```
public Etudiant(String nom, int annee){
   nom = this.nom ;
   annee = this.annee ;
}
```

   Qu'est-ce que fait ce code ?

Exercice II

Dans cet exercice nous allons utiliser notre constructeur pour créer des objets de type Etudiant.

1. Petit rappel :
    - Qu'est-ce qu'une méthode principale (main) ?
    - Comment on définit une telle méthode dans une classe ?
    - Comment peut-on initialiser des objets de type Etudiant dans une autre classe ?
    - Nomenclature : comment écrit-on des

2. Dans une nouvelle classe MonTD1, dans la méthode principale, créez ces objets :

    - Un étudiant de première année avec le nom « `Jean Dupont` ».
    - Une étudiante de deuxième année avec le nom « `Marine Delafontaine` »

3. La plupart de nos étudiants vont être de première année. Pour simplifier leur initialisation, on écrit un deuxième constructeur dans la classe `Etudiant`, avec la signature `Etudiant(String nom)`. Ce constructeur initialise l'attribut `nom` à la valeur en entrée, et initialise l'attribut `annee` à la valeur par défaut `1`.
   En général, c'est une bonne pratique d'appeler le constructeur avec plus de paramètres dans celui avec moins de paramètres. On appelle le premier constructeur à partir du deuxième en utilisant le mot dédié `this`.

- Pourquoi est-ce que c'est une bonne pratique de faire cela ?
- Ecrivez le code du deuxième constructeur.
- Comment peut-on remplacer le code qui crée l'étudiant au nom de Jean Dupont en utilisant ce nouveau constructeur ?

Exercice III

Dans cet exercice nous allons ajouter des méthodes pour qu'on puisse avoir accès aux attributs des objets de type `Etudiant`.

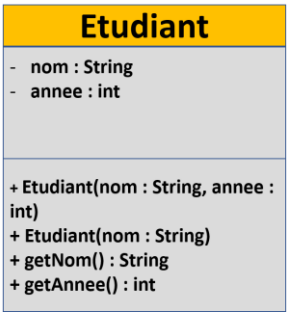
1. Disons que dans notre méthode principale nous avons le code suivant :

```
final thibaultLagrange = new Etudiant("Thibault Lagrange");
System.out.println(thibaultLagrange.annee);
```

   - Est-ce que ce code compile ?
   - Qu'est-ce qui se passe lors de l'exécution ?

2. Pour avoir accès aux attributs de la classe `Etudiant` en dehors de cette classe on pourrait : a) faire les attributs publiques ; b) ajouter des méthodes de type `getAttribut()` qui rendront les valeurs stockées par les attributs en question.
   - Discutez les avantages et les inconvénients de ces deux méthodes
   - Petit rappel : quelle est la syntaxe d'une méthode qui rend une valeur à la sortie ? Qu'en des méthodes qui ne rendent aucune valeur ?
   - Ecrivez, dans la classe `Etudiant`, deux méthodes qui rendent les valeurs des attributs de la classe Etudiant, avec les signatures :
     - `String getNom()`
     - `int getAnnee()`

   Ceci changera le diagramme de classe de la classe `Etudiant` à :



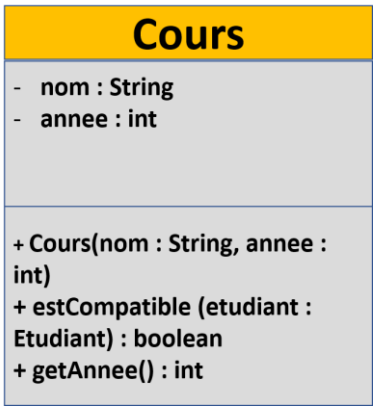
3. Modifiez le code au point 1. tel qu'il finit par afficher l'année de l'étudiant `thibaultLagrange`.

Exercice IV

Dans cet exercice nous allons créer une nouvelle classe, `Cours`, dont les objets seront des cours que les étudiants pourront choisir pour leur programme d'étude, à condition que le cours soit compatible avec leur année.

1. Nous allons mettre en place une classe `Cours` avec deux attributs et trois méthodes :
    - Un attribut `nom` de type `String`
    - Un attribut `annee` de type entier
    - Un constructeur avec la signature `Cours(String nom, int annee)`, qui mettra les attributs de l'instance de `Cours` aux valeurs en entrée
    - Une méthode `boolean estCompatible(Etudiant etudiant)` qui retourne une valeur de type `boolean` : `true` si l'étudiant a la même `annee` que le cours, et `false` autrement
    - Une méthode `int getAnnee()` qui retourne la valeur actuellement stockée par l'attribut `annee` de l'instance de `Cours`.

    Un résumé de cette classe est capturé par le diagramme de classe ci-dessous :



    - Ecrivez la méthode `estCompatible(Etudiant etudiant)`

2. Nous allons ensuite modifier la classe Etudiant : on y rajoute un attribut qui s'appelle mesCours et qui est de type Cours[]. Un étudiant pourra choisir jusqu'à un maximum de 20 cours.
    - Petit rappel : comment initialise-t-on un tableau dans un constructeur ?

- Réécrivez les constructeurs de la classe Etudiant pour prendre en compte ce nouveau tableau.

3. Nous voulons pouvoir ajouter un nouveau cours aux cours d'un étudiant. À ce fin nous voulons écrire une méthode void ajouterCours(Cours cours) qui :
   a. Cherche si le cours est compatible avec l'étudiant en question (this) et, le cas échéant,
   b. Cherche la première position non-occupée du tableau, pour
   c. Y ajouter le cours.

   - Voici une partie du code de cette méthode. Expliquez ce qu'elle fait.

```
public void ajouterCours(Cours cours) {
	if (cours.estCompatible(this)) {

		boolean ajoute = false;
		int positionActuelle = 0;

		while (!ajoute && positionActuelle < mesCours.length) {
			if (null == this.mesCours[positionActuelle]) {
				//A COMPLETER
			}
			// A COMPLETER
		}
		if (!ajoute) {
			// A COMPLETER
		}
	}
}
```

   - Complétez ce code.

4. Ecrivez du code (à mettre dans la méthode principale) qui :
   - Crée un cours qui s'appelle "Java" de première année, et puis
   - Permet à l'Etudiant appelé "Jean Dupont" d'ajouter ce cours.