M2103-- TD1

Introduction aux TDs en Java

Bonne pratique : toujours réfléchir 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

Préambule

Nous ferons premièrement quelques exercices pour mieux se rappeler de comment on travaille en Java.

Nous allons partir sur une classe Nombre, partant sur le code suivant :

```
public class Nombre {
	int nombre;

	public Nombre(int nombre) {
		this.nombre = nombre;
	}

	//methode pour calculer nombre + y
	public Nombre ajouter(Nombre y) {
		int valeur = this.nombre + y.nombre;
		return new Nombre(valeur);
	}

	//methode pour calculer nombre|y
	public String concatener(Nombre y) {
		return (String.valueOf(this.nombre) + String.valueOf(y.nombre));
	}

}
```

1. Petit rappel :
    - Qu'est-ce qu'un attribut ?
    - Qu'est-ce qu'un constructeur ?
    - Qu'est-ce qu'une variable ou méthode publique ? Et une variable privée ?

2. Trouvez le constructeur dans cette classe. Qu'est-ce qu'il fait ?

3. Expliquez l'utilité de la deuxième instruction de la méthode ajouter. Pourquoi est-elle nécessaire ?

4. 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 les noms des variables ?

5. Dans une deuxième classe `JeuDeNombres`, nous avons le code :

```
public class JeuDeNombres {

	public static void main(String[] args) {

		final Nombre douze = new Nombre(12);

	}

}
```

   Qu'est-ce que fait ce code ?

6. Nous ajoutons la méthode suivante dans le code de la classe Nombre :

```
public String toString() {
	System.out.println(this);
	return "-1";
}
```

   Disons que dans la méthode `main` de la classe `JeuDeNombres` on rajoute le code suivant :

```
System.out.println(douze);
```

   Est-ce que le code compile ? Quel est le résultat de l'exécution ?

7. Nous remplaçons le code de la méthode `toString` par le code suivant :

```
public String toString() {
	return "Nombre[" + this.nombre + "]";
}
```

Répondez aux questions de l'exercice précédent pour ce nouveau code.

8. Nous ajoutons les méthodes suivantes :

```
public Nombre ajouterUn() {
	Nombre un = new Nombre(1);
	return this.ajouter(un);
}

public Nombre ajouterUnBis() {
	Nombre un = new Nombre(1);
	return un.ajouter(this);
}

public String concatenerUn() {
	Nombre un = new Nombre(1);
	return (this.concatener(un));
}

public String concatenerUnBis() {
	Nombre un = new Nombre(1);
	return un.concatener(this);
}
```

Dans la méthode `main` de la classe `JeuDeNombres`, nous rajoutons le code suivant sous l'instruction de l'exercice antérieur :

```
System.out.println(douze.ajouterUn());
System.out.println(douze.ajouterUnBis());
System.out.println(douze.concatenerUn());
System.out.println(douze.concatenerUnBis());
```

Quel est le résultat de ce code ?

Exercice I

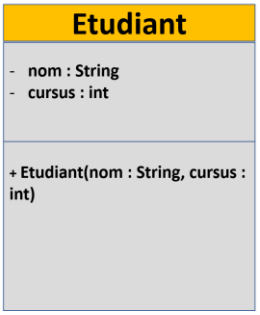
On commence par une classe `Etudiant`. Dans celle-ci nous aurons les prochains attributs :

- Un attribut nom de type `String` ;

- Un attribut `cursus` de type entier ;

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

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

2. Imaginons que le code suivant remplace le constructeur que vous venez d'écrire.

```
public Etudiant(String nom, int cursus){
   nom = this.nom ;
   cursus = 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. Nous allons utiliser une nouvelle classe TD1 qui aura une méthode principale (`main`). Dedans, nous allons créer des objets de type `Etudiant`.

1. Regardez le constructeur que vous avez créé dans l'exercice I. Quelles variables (type et nom de variable) demande-t-il en entrée ?

2. Dans une nouvelle classe `TD1`, 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 `cursus` à la valeur par défaut `1`.

    Ecrivez ce constructeur.

4. En général, c'est une bonne pratique d'appeler le constructeur avec plus de paramètres dans celui avec moins de paramètres. Lorsqu'on appelle le constructeur avec plus de paramètres, nous pouvons remplacer quelques variables en entrée par des constantes, en fonction des spécifications du constructeur.

    Pour appeler le premier constructeur à partir du deuxième en utilisant le mot dédié `this(…)` -- en paranthèse les paramètres demandés par le constructeur appelé.

    - Pourquoi est-ce que c'est une bonne pratique de faire cela ?
    - Comment peut-on remplacer le code écrit à l'exercice précédent en utilisant le constructeur avec plus de paramètres dans celui avec moins de paramètres ?
    - Comment peut-on remplacer le code qui crée l'étudiant au nom de Jean Dupont (dans la méthode `main` de la classe `TD1`) en utilisant ce nouveau constructeur ?
    - Peut-on utiliser le deuxième constructeur pour créer l'étudiant au nom Marine Delafontaine ?

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 :

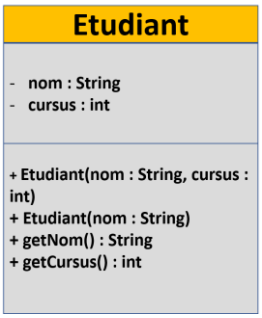
```
final Etudiant thibaultLagrange = new Etudiant("Thibault
Lagrange");
System.out.println(thibaultLagrange.cursus);
```

Analysez chaque ligne en regardant les aspects suivants :

- 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 getCursus()`

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



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