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 L'affichage et les instructions de base

Consignes générales

Dans ce TD vous allez travailler en groupes de 4 étudiants. Vous pouvez choisir votre groupe, et votre enseignant vous attribuera un nombre (de 1 à 6).

Ensuite, vous allez résoudre les exercices 1 à 5 ci-dessous en groupe. Vous aurez 50 minutes.

À la fin de l'exercice 5, vous allez avoir un code de base pour 3 classes. Chaque membre du groupe de 4 personnes doit avoir écrit une copie de cette solution sur un papier. À ce point-là, vous allez échanger vos solutions comme ci-dessous. C'est une bonne pratique de programmation que l'on appelle **revue de code.**

Groupe 1 échange avec Groupe 2

Groupe 3 échange avec Groupe 4

Groupe 5 échange avec Groupe 6

Chaque groupe corrige le travail de l'autre groupe. Vous allez vous concentrer sur :

- Le fonctionnement du code – est-ce que le code compile ? Est-ce qu'il produit un bon résultat ?
- Le suivi des consignes : est-ce que les classes ont été définies en suivant les instructions ci-dessous ?
- La lisibilité du code : est-ce que le code est facile à suivre ?
- L'efficacité du code : est-ce qu'on peut améliorer l'efficacité du code d'un façon ou d'autre ?

Vous aurez 30 minutes.

Finalement, chaque membre de chaque groupe explique ses corrections à un membre de l'autre groupe. Vous aurez 15 minutes.

1. Définir une classe `Serie`, dont les objets auront comme attributs : le `premier élément` de la série, l'`écart` entre deux éléments, et le nombre total d'éléments dans la série.

   Définir un constructeur qui instancie les objets de type Serie, avec la signature :

   Serie(int premierElement, int ecart, int nombreElements)

2. Définir une méthode **toString()** qui affiche tous les éléments d'une serie (serie = un objet de la classe Serie, d'où le manque d'accent). Les éléments seront affichés en ordre sur une seule ligne, avec un écart de deux caractères de type espace entre chaque élément.

   Par exemple la série contenant les éléments 0 1 2 3 4 5 sera affiché comme ci-dessous :

   0 1 2 3 4 5

3. Nous souhaitons générer et afficher maintenant une table de Pythagore, notamment une table de multiplication de taille arbitraire. Pour une taille de 5, ceci sera la table de Pythagore à afficher :

   ```
   *  0  1  2  3  4  5
   0  0  0  0  0  0  0
   1  0  1  2  3  4  5
   2  0  2  4  6  8 10
   3  0  3  6  9 12 15
   4  0  4  8 12 16 20
   5  0  5 10 15 20 25
   ```

   Attention à l'alignement !
   Pour réaliser cette table de Pythagore, suivez les instructions ci-dessous.

   Nous allons noter chaque ligne par son premier symbole à gauche. Par exemple le première ligne sera, la «Ligne *», tandis que la troisième ligne sera la «Ligne 1». Dans chaque ligne, chaque élément i (à partir du deuxième élément) est le produit du premier élément de la ligne avec l'élément i de la Ligne *.

   Nous allons définir la classe TablePythagoras avec un seul attribut taille de type entier. Dans le constructeur d'un TablePythagoras la taille sera initialisée.

4. Définir une méthode toString(), qui affiche une table de Pythagoras. Nous allons utiliser l'instruction +"\n" pour revenir à la ligne.
   Par exemple le code :

String myString = "2"+ "\n" + "3";
System.out.println(myString);

affichera

2
3

Pour la méthode toString() de la classe TablePythagoras nous allons utiliser la méthode toString() de la classe Serie. Ainsi, chaque ligne sera considérée comme une série. Pour écrire la méthode toString() de la classe TablePythagoras, réfléchir en premier aux prochaines questions :

- Quel sera le résultat du code suivant par rapport à l'alignement des deux lignes ?

```
Serie serie1 = new Serie(0, 1, 6);  // série avec premElem =0, ecart =1, noelem =6
Serie serie2 = new Serie(10, 5, 6); //série avec premElem =10, ecart =5, noelem =6
System.out.println(s1);
System.out.println(s2);
```

- Pourquoi ?
- Comment peut-on assurer l'alignement de ces deux lignes ?
- Comment faut-il modifier le toString() de la classe Serie pour toujours assurer l'alignement de deux séries ?
- Si nous ignorons le premier élément de chaque Ligne, comment dans ces conditions, peut-on encore voir une ligne comme une série ?
- Finir d'écrire la méthode toString() de la classe TablePythagoras.

5. Définir une classe principale AffichageTable, qui n'a aucun attribut. Dans la méthode main(String[] args) nous allons utiliser args[0] pour stocker la taille de la table de Pyhtagoras qu'on veut afficher.

- Quel type d'élément est args[0] ?
- Est-ce que le code suivant compile ? Sinon, pourquoi pas ?

```
TablePythagoras maTable = new TablePythagoras(args[0]) ;
```

- Une instruction très utile, qui convertit un String en entier est l'instruction Integer.parseInt(<varString>). En utilisant cette instruction, corrigez le code ci-dessus pour le faire compiler.
- Écrire votre classe AffichageTable.