TP 3 - Travail préliminaire

*Travail à réaliser **avant** la séance de TP*

Le but de cette première partie est :

▷ de découvrir les outils `javac` et `java`

▷ de découvir la « javadoc »[1] et la structure de la documentation d'une classe

A partir de ce TP on abandonne BlueJ.

Utilisez un éditeur adapté à la programmation (Atom, Visual Studio Code, Emacs ou ... car ils ont l'avantage de disposer d'un « mode java » avec coloration syntaxique et qui facilite le travail d'indentation).
Faites attention de bien veiller à sauvegarder votre code source avant de le compiler.

Exercice 1 : Premières compilation et exécution
Nous allons travailler dans cet exercice avec la classe `Stock` du TP de la semaine passée (une version est disponible sur le portail).
Nécessairement le code source de cette classe se trouve dans un fichier qui s'appelle `Stock.java`. En JAVA il y a toujours correspondance entre le nom du fichier et le nom de la classe qu'il définit, et l'extension est nécessairement `.java`.

Q 1 . Dans un terminal, placez-vous dans le répertoire contenant votre fichier `Stock.java`. Effacez le fichier `Stock.class` qui s'y trouve probablement.

Pour compiler votre fichier, il faut exécuter la commande :

```
javac Stock.java
```

Nous verrons dans une prochaine séance de TP que les choses sont parfois un peu plus complexes pour la compilation.

Faites le. S'il n'y a pas eu d'erreur de compilation, votre répertoire contient maintenant le fichier `Stock.class` généré par la compilation. C'est ce fichier qui contient le bytecode JAVA utilisé par la machine virtuelle.

Q 2 . Exécuter un programme JAVA consiste à exécuter le corps d'une méthode particulière placée dans une classe. Cette méthode a **obligatoirement** et **rigoureusement** la signature :

```
public static void main(String[] args)
```

Il peut y avoir (au plus) une méthode avec cette signature par fichier (définissant une classe). L'exécution d'une telle méthode contenue dans une classe `SomeClass` est provoquée par la commande (il faut que vous soyez dans le répertoire contenant le fichier `SomeClass.class`) :

```
java SomeClass
```

A nouveau, les choses seront nuancées dans un prochain TP, notamment du fait des paquetages.

Cette commande « démarre » une machine virtuelle JAVA (JVM) qui exécute les lignes de code contenues dans la méthode `main` de la classe passée en argument (ici `SomeClass`). Le paramètre `args` de cette méthode contient le tableau des éventuels autres[2] arguments ajoutés à la ligne de commande.

Notez qu'il n'y a pas d'extention « `.java` » (ni « `.class` » d'ailleurs), ici on exécute une classe alors qu'avec `javac` on compilait un fichier (ce qui a permis de définir la classe). Pour cette commande, il est donc nécessaire de disposer du fichier `SomeClass.class`, mais pas nécessairement du source (`.java`).

Ajoutez maintenant à votre fichier `Stock.java` la définition :

```
public static void main(String[] args) {
   Stock someStock = new Stock();
   someStock.add(10);
   System.out.println(someStock.getQuantity());
}
```

[1] La documentation des API java, s'appelle la « javadoc » du nom de l'outil qui sert à les générer et que nous aborderons dans un prochain TP.

[2] A la différence de `python` où le nom du script est repris dans la liste `sys.argv`, le nom de la classe lui n'est pas repris dans ce tableau en java.

Sauvez, compilez la classe `Stock` puis exécutez-en la méthode `main` à l'aide de la commande `java Stock`.

Q 3 . Modifiez la méthode `main` de la classe `Stock` comme indiqué ci-dessous (la différence se situe au niveau du paramètre de la méthode `add()`, les explications sont données après le code) :

```
public static void main(String[] args) {
   Stock someStock = new Stock();
   someStock.add(Integer.parseInt(args[0]));
   System.out.println(someStock.getQuantity());
}
```

Commentaires :

- "`Integer.parseInt`" prend en paramètre une chaîne de caractères (donc un objet de type `String`) et renvoie un `int` correspondant au contenu de cette chaîne si elle représentait un entier.
  exemple : `Integer.parseInt("42")` vaut l'`int 42`.
  Il s'agit de l'utilisation de la méthode **static** de la classe `Integer` :

  `public static int parseInt(String s)`
- `args[0]` est le premier argument qui est fourni après le nom de la classe dans la ligne de commande, pas le nom de la classe lui-même. Cette valeur est touours une chaine de caractères.
  `Integer.parseInt(args[0])` permet donc de traduire en une valeur entière le premier argument passé en ligne de commande lors de l'exécution du programme, par exemple : `java Stock 12`.
- de manière similaire avec ce que vous avez peut-être déjà pris l'habitude de faire en TP de Python, on peut ajouter les lignes suivantes au tout début de la méthode `main`, pour informer sur l'usage des arguments :

```
if (args.length < 1) {
   System.out.println("usage :  java Stock <unEntier>");
   System.exit(0);  // arrête l'exécution
}
```

Q 4 . Sauvez ce fichier, compilez-le puis exécutez sa méthode `main` en ajoutant à la commande un argument supplémentaire qui devra correspondre à un entier.

Par exemple : `java Stock 12`.

Exercice 2 : La JavaDoc de l'API Java

En ouvrant dans votre navigateur l'url

`http://docs.oracle.com/javase/8/docs/api/`

vous visualisez l'ensemble de la JavaDoc des paquetages fournis en standard avec le jdk[3].

En haut à gauche se trouve la liste des paquetages, en dessous la liste des classes du paquetage sélectionné (initialement toutes les classes) et dans la partie de droite la documentation de la classe sélectionnée (initialement la liste des paquetages).

Q 1 . Dans la liste des paquetages (en haut à gauche) sélectionnez le paquetage `java.lang`

Q 2 . Dans la liste des classes (en bas à gauche) sélectionnez la classe `String` et parcourez rapidement sa documentation :
Dans la zone "description de classes" (cadre de droite), la documentation est toujours organisée selon la même structure :

1. description de la classe
2. résumés :
   (a) les attributs (seuls ceux qui seraient publics appparaitraient)
   (b) les constructeurs (publics)
   (c) les méthodes (publiques)
3. détails :
   (a) les attributs : description
   (b) les constructeurs : présentation et description des paramètres
   (c) les méthodes : présentation, description des paramètres et des valeurs de retour

Trouvez et lisez les documentations des méthodes `charAt()`, `length()` et `substring()` de la classe `String` et qui avaient été évoquées dans le document du premier TD.

Identifiez les desccriptions du rôle de la méthode, des paramètres et de la valeur de retour.

[3] Java Development Kit.