TP 3 - Partie 1

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 (Emacs, Gedit, SciTe et Kate ont l'avantage de disposer d'un "mode java" facilitant entre autres le travail d'indentation).

Exercice 1 : Premières compilation et exécution
Nous allons travailler dans cet exercice avec la classe `Stock` du TP de la semaine passée.
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 une commande shell, 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. Si il n'y a pas 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 `UneClasse` est provoquée par la commande (il faut que vous soyez dans le répertoire contenant le fichier `Stock.class`) :

```
java Stock
```

A nouveau, les choses seront nuancées dans le prochain TP.

Cette commande "lance" une machine virtuelle JAVA (JVM) qui exécute les lignes de code contenues dans la méthode `main` de la classe passée en argument (ici `Stock`). 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 de "`.java`", ici on exécute une classe alors qu'avec `javac` on compile un fichier (qui permet de définir une classe). Pour cette commande, il est donc nécessaire de disposer du fichier `UneClasse.class`, mais pas nécessairement du source (`.java`).

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

```
public static void main(String[] args) {
   Stock unStock = new Stock();
   unStock.ajoute(10);
   System.out.println(unStock.getQuantite());
}
```

Sauvez, compilez puis exécutez.
NB : on peut ajouter les lignes suivantes au tout début de la méthode `main`, pour informer sur l'usage des arguments :

[1] La documentation des API java, s'appelle la "javadoc" du nom de l'outil qui sert à les générer et que nous aborderons une autre fois.
[2] A la différence de `ocaml` où le nom du programme est repris dans le tableau d'arguments, le nom de la classe lui n'est pas repris dans ce tableau en java.

```
if (args.length <= 0) {
   System.out.println("usage :  java Stock <unEntier>");
   System.exit(0);
}
```

Q 3 . Modifiez ainsi la méthode `main` de la classe `Stock` :

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

Commentaires :

- "`Integer.parseInt`" prend en paramètre une chaîne (objet de type `String`) et renvoie un `int` correspondant à cette chaîne si elle représente 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 de la ligne de commande.

Sauvez, compilez puis exécutez avec 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 le fichier `/opt/java/jdk1.6.0_26/docs/api/index.html` vous visualisez l'ensemble de la JavaDoc des paquetages fournis en standard avec le jdk[3] (**Ajoutez cette page à vos signets/marque-pages. En salle de TP, un lien direct est disponible dans la rubrique "Documents" sur le portail.**).

Oui, cette documentation est en anglais.
Non, il n'existera pas de version française. Il faut absolument vous habituer à lire et comprendre de la documentation technique en anglais.

En haut à gauche se trouve la liste des paquetages (notion présentée la semaine prochaine), 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)
   (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

[3] Java Development Kit.