TP 1 (partie 1) : environnement Java, utilisation de quelques classes

Exercice 1 : documentation en ligne

La documentation JAVA se trouve à l'adresse :
https://docs.oracle.com/en/java/javase/11/docs/api/

Question 1 : structure de l'API
L'API Java (version 11 ici) est constituée de classes, d'interfaces, d'exceptions, etc. qui sont regroupées dans des **paquetages**, les paquetages étant eux-mêmes regroupés dans des **modules**.

La documentation présente d'abord une vue générale (*Overview*) de tous les **modules** qui constituent l'API.

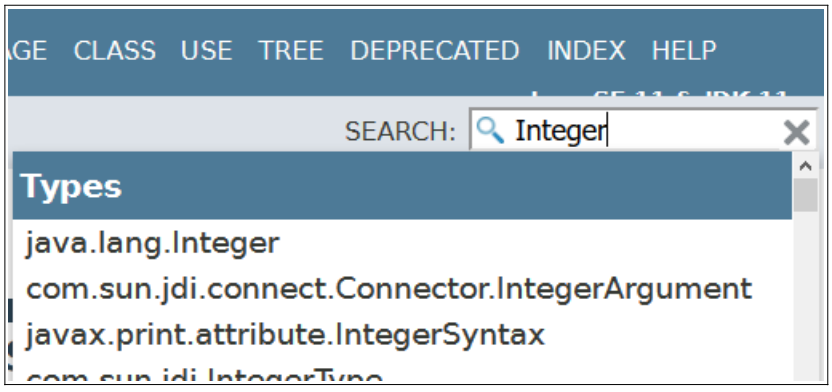
> *Accédez à la documentation en ligne. Cliquez sur le module* **`java.base`** *(c'est sans soute le seul qui vous intéressera au cours du semestre 2). Vous accédez alors à tous les paquetages de ce module.*

> *Cliquez ensuite sur le paquetage* **`java.lang`** *(qu'on utilisera souvent) et regardez son contenu. De quoi est-il composé ?*

Question 2 : rechercher une classe
Pour rechercher une classe dans la documentation, on peut :

- soit sélectionner le paquetage qui la contient puis sélectionner la classe dans ce paquetage (mais encore faut-il connaître ce paquetage),

- soit utiliser la barre de recherche, sur la droite de la documentation :



Recherche de la classe Integer à l'aide de la barre de recherche

Question 3 : documentation d'une classe

La documentation d'une classe commence par le rappel de ses caractéristiques générales :

puis indique sa déclaration :

```
public final class Integer
extends Number
implements Comparable<Integer>
```

puis décrit à quoi elle sert :

The `Integer` class wraps a value of the primitive type `int` in an object. An object of type `Integer` contains a single field whose type is `int`.

In addition, this class provides several methods for converting an `int` to a `String` and a `String` to an `int`, as well as other constants and methods useful when dealing with an `int`.

et enfin présente trois parties, d'abord sous forme synthétique (*summary*) puis sous forme détaillée :

- **Fields** : les variables (de classe ou d'instance) de la classe,
- **Constructors** : le ou les constructeurs de la classe,
- **Methods** : les méthodes (de classe ou d'instance) de la classe.

Comment savoir à quoi sert le package `java.io` ?

Dans le paquetage `java.lang`, à quoi sert la classe `System` ?

Question 4
Dans l'instruction :

```
System.out.println("Bonjour");
```

- Pourquoi le mot `System` a-t-il une majuscule ?
- Que représente le mot `out` ? De quel type est-ce ?
- Que représente le mot `println` ?

Question 5
Dans l'instruction :

```
Scanner entree=new Scanner(System.in);
```

- Que représente le mot `Scanner` ?
- Comment trouver rapidement les informations sur cette classe dans la documentation en ligne ?
- `in` est-elle une variable de classe ou d'instance ? De quel type est cette variable ?

Exercice 2 : compilation et exécution

La compilation sous l'invite Unix se fait par la commande *javac* :

```
javac  monProgramme.java
```

Le fichier à compiler doit avoir l'extension `.java`. Son exécution se fait par la commande *java* :

```
java  monProgramme
```

Vous pouvez saisir le code source d'une application JAVA avec n'importe quel éditeur de texte.

Question 1
Écrivez un programme qui affiche la chaîne de caractères *bonjour* à l'écran.

Question 2
Compilez ce programme. Regardez le contenu de votre répertoire. Que constatez-vous ?

Question 3
Exécutez votre programme.

Exercice 3 : la classe String

Le but de cet exercice est de vous faire explorer la classe `String` du paquetage `java.lang` en testant ses méthodes sur des chaînes et d'autres valeurs lues au clavier, et de vous initier à la consultation de la documentation en ligne. D'une façon générale, tout langage de programmation comporte des méthodes diverses et variées sur les chaînes de caractères. Sachez repérer dans la documentation, la méthode qui répond à vos besoins.

Dans cet exercice, vous devez créer votre propre classe `TestChaines` dont la méthode principale `public static void main (String[] args)` effectue les actions demandées dans les questions suivantes.

Question 1
Créez une variable de type `int`, affectez-lui une valeur, puis convertissez cette variable en chaîne (ex.: le nombre 12345 devient la chaîne "12345"). Utilisez la méthode `valueOf()` de la classe `String`.

Question 2
Au clavier, lire une chaîne entièrement composée de chiffres et la convertir dans le nombre entier qu'elle représente (ex.: la chaîne "12345" devient le nombre 12345). La solution se trouve parmi les méthodes statiques de la classe `java.lang.Integer`. Après affichage du nombre entier, vous lui ajouterez 1 et afficherez le nouveau nombre obtenu. Utilisez la méthode `parseInt()` de la classe `Integer`.

Question 3
Même question que ci-dessus, mais avec un nombre flottant (ex.: la chaîne "0.12345e4"devient le nombre 0.12345e4). Après affichage du nombre, vous lui ajouterez 1.1 et afficherez le nouveau nombre obtenu. Utilisez la méthode `parseFloat()` de la classe `Float`.

Question 4
Lire une chaîne représentant un nom de ville, lui enlever les éventuels blancs au début et à la fin et l'afficher entièrement en majuscules.

Question 5
Lire deux chaînes `s1` et `s2` et afficher la réponse à la question: «ces deux chaînes commencent-elles par le même caractère?» Utilisez la méthode d'instance `charAt`.

Question 6
Lire deux chaînes `s1` et `s2` et afficher les résultats renvoyés par les expressions :

- `s1==s2`,
- `s1.equals(s2)`,
- `s1.compareTo(s2)`,
- et `s1.compareToIgnoreCase(s2)`.

Entre autres, essayez les couples "abcd" et "abcd", puis "abcd" et "AbcD".

Question 7
Lire deux deux chaînes `s1` et `s2` et afficher la réponse aux questions :

- `s1` commence-t-elle par `s2` ?
- `s1` finit-t-elle par `s2` ?
- `s1` contient-elle `s2` ?

Vous utiliserez les méthodes `startsWith()`, `endsWith()` et `contains()`.

Question 8
Lire deux chaînes `s1` et `s2` et, si `s1` contient `s2`, afficher `s1` privée de `s2` sinon afficher `s1`. Intéressez-vous à `substring` et `indexOf`.

Exercice 4 : une calculatrice trigonométrique

Le but de cet exercice est de réaliser une petite calculatrice qui calcule le cosinus, le sinus ou la tangente d'un angle. Vous utiliserez les méthodes de classe de la classe `Math` (paquetage `java.lang`) pour effectuer les calculs : méthodes de classe cos(double a), sin(double a) et tan(double a) dont le résultat est de type primitif `double` et dont l'argument est la mesure d'un angle en radian. Consultez la documentation.

Votre programme devra boucler sur le menu suivant. L'utilisateur choisira un calcul dans ce menu (C, S ou T) puis saisira la valeur d'un angle, en radians.

```
C : calcul d'un cosinus
S : calcul d'un sinus
T : calcul d'une tangente
Q : quitter le programme
```

Après la saisie de la valeur de l'angle (sous forme de `String`), il faudra la convertir en type primitif `double`.

Voici quelques méthodes nécessaires aux conversions :

- la méthode de classe `valueOf(String str)` de la classe `Double` a comme argument une instance de classe `String` et délivre la conversion de la chaîne en instance de `Double`,
- la méthode d'instance `doubleValue()` délivre la conversion d'une instance de `Double` en type primitif `double`.