TP JAVA

1. Exercices simples pour commencer

1.1. L'incontournable « Print "Hello" »

Saisissez, compilez et exécutez un programme Java qui affiche *Bonjour à tous !* sur l'écran de la console. Écrivez une classe publique exécutable nommée **`Bonjour`** placée dans un fichier nommé **`Bonjour.java`**.

Trois « détails » à ne pas oublier :

1. Une classe est exécutable si et seulement si elle comporte une méthode ayant exactement la signature « **`public static void main(String[]`** *args***`)`** »
2. On provoque une compilation en donnant un nom *de fichier* (ex.: **`javac Bonjour.java`**)
3. On lance l'exécution d'un programme en donnant un nom *de classe* (ex.: **`java Bonjour`**).

1.2. Les arguments de la ligne de commande

Dans sa forme générale, la commande qui déclenche l'exécution d'une application Java s'écrit

`java` *UneClasse chaîne*$_0$ *chaîne*$_0$ ... *chaîne*$_{k-1}$

où *chaîne*$_0$ *chaîne*$_0$ ... *chaîne*$_{k-1}$ sont des chaînes de caractères ne contenant pas de blancs séparées par des blancs (espaces, tabulations).

Ces chaînes sont accessibles depuis l'intérieur de l'application à travers le tableau, généralement appelé **`args`**, qui est déclaré dans l'en-tête de la fonction **`main`**. Par exemple, si une application est lancée par la
commande **`java NomDeLaClasse Dupond "Jean Pierre" 2006`**

`2+3=5`

alors la fonction **`main`** reçoit un argument qui est un tableau formé des *quatre* chaînes de caractères **`"Dupond"`**, **`"Jean Pierre"`**, **`"2006"`** et **`"2+3=5"`**. Notez que :

- quand *un* argument contient des blancs (Jean Pierre) on doit l'encadrer par des guillemets,
- les nombres ne sont pas reconnus en tant que tels, mais en tant que chaînes de caractères,
- les seuls séparateurs reconnus à ce niveau sont l'espace et la tabulation ; par exemple, **`2+3=5`** forme un seul argument.

Exercice. Écrivez une classe **`Calcul`** qu'on exécute avec trois arguments : deux nombres et un opérateur entre les deux, et qui affiche le résultat de l'opération indiquée sur les nombres indiqués. Exemple d'exécution (ce qui est saisi par l'utilisateur est bleu) :

```
java Calcul 1200 + 400
1200 + 400 = 1600
```

1.3. Jeux avec les chaînes

But de cet exercice : vous faire explorer la classe **`java.lang.String`** en testant ses diverses méthodes sur des chaînes et d'autres valeurs lues au clavier.

> Un des buts de cet exercice est justement de vous initier à la consultation de cette documentation. Pour cela, ouvrez un navigateur sur la doc de l'API (chez Sun, http://docs.oracle.com/javase/7/docs/api/) ; dans le volet gauche supérieur sélectionnez le paquetage **`java.lang`** et, alors, dans le volet gauche inférieur sélectionnez la classe **`String`**.

Il s'agit d'écrire une classe exécutable **`TestChaines`** dont la méthode principale effectue les opérations suivantes :

A. Créez une variable de type **`int`**, affectez-lui une valeur, puis convertissez cette variable en chaîne (ex.: le nombre **`12345`** devient la chaîne **`"12345"`**).

B. Sur la ligne de commande, 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`**.

C. Même question que ci-dessus, mais avec un nombre flottant (ex.: la chaîne **`"0.12345e4"`** devient le nombre **`0.12345e4`**).

D. 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.

E. Lire deux chaînes s_1 et s_2 et afficher la réponse à la question : « ces deux chaînes commencent-elles par le même caractère ? » Utilisez la méthode d'instance **`charAt`**.

F. Lire deux chaînes s_1 et s_2 et afficher les résultats renvoyés par les expressions « s_1 **`==`** s_2 », « s_1**`.equals(`**s_2**`)`** », « s_1**`.compareTo(`**s_2**`)`** » et , « s_1**`.compareToIgnoreCase(`**s_2**`)`** ». Entre autres, essayer les couples **`"abcd"`** et **`"abcd"`**, puis **`"abcd"`** et **`"AbcD"`**.

G. Lire deux chaînes s_1 et s_2 et afficher la réponse aux questions « s_1 commence-t-elle par s_2 ? », « s_1 finit-elle par s_2 ? » et « s_1 contient-elle s_2 ? »

H. Lire deux chaînes s_1 et s_2 et si s_1 contient s_2, renvoyer s_1 privée de s_2 (s'intéresser à **`substring`**), sinon renvoyer s_1.

I. Si *s* est une chaîne de caractères, l'expression *s***`.intern()`** renvoie une chaîne ayant les mêmes caractères que *s* mais appartenant à une collection de chaînes *sans doublons* que Java maintient et dont les chaînes figurant dans le programme source font partie d'office.

Lire deux chaînes **`s1`** et **`s2`** et constater qu'à la suite des transformations « **`s1 = s1.intern();`** » et « **`s2 = s2.intern();`** » les expressions « **`s1.equals(s2)`** » et « **`s1 == s2`** » deviennent équivalentes.

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.

1.4. Random

Pour tester la méthode **`java.lang.Math.random()`**, écrivez un programme qui obtient *n* nombres « au hasard » (par exemple, *n* = 100 000) et qui calcule la *moyenne* et l'*écart-type* de la suite obtenue.

Pour produire des nombres aléatoires utilisez la méthode **`java.lang.Math.random()`** (cela se lit : « la méthode **`random`** de la classe **`Math`** du paquetage **`java.lang`** »), sur laquelle vous trouverez des indications dans la documentation de l'API.

> Un des buts de cet exercice est justement de vous initier à la consultation de cette documentation. Pour cela, ouvrez un navigateur sur la doc de l'API (chez Sun, http://docs.oracle.com/javase/7/docs/api/) ; dans le volet gauche supérieur sélectionnez le paquetage **`java.lang`** et, alors, dans le volet gauche inférieur sélectionnez la classe **`Math`**. Parmi les méthodes de cette classe, cherchez **`random`** et lisez-en la documentation.

Rappel de formules : si *n* est le nombre de termes d'une suite, $S = x_1 + x_2 + ... + x_n$ la somme de ces termes et $Q = x_1^2 + x_2^2 + ... + x_n^2$ la somme de leurs carrés, alors

- la moyenne est $m = S / n$
- l'écart-type *s* est la racine carrée de la variance *V*, donnée par $V = (Q / n) - m^2$

N.B. Si la distribution des valeurs renvoyées par la fonction `random` est uniforme, quand *n* devient de plus en plus grand *m* doit tendre vers 0,5 et *s* vers 0,288675134595 (la racine carrée de 1/12).

1.5. Calcul de la factorielle

A. Écrivez une classe exécutable avec une méthode :

```
static long factorielle1(int n)
```

qui calcule la factorielle de *n*, c'est-à-dire le nombre $n ! = n \times (n - 1) \times (n - 2) \times ... \times 3 \times 2 \times 1$.

B. Trouvez la valeur de *n* à partir de laquelle les débordements des valeurs entières rendent cette méthode inutilisable.

N.B. Une manière de vérifier que la valeur de $n!$ est juste, sachant que celle de $(n - 1)!$ l'est, consiste à afficher le rapport $n! / (n - 1)!$

C. Pour pallier ce problème de débordement, écrivez une nouvelle fonction utilisant des objets**`java.math.BigInteger`** :

```
static BigInteger factorielle2(int n)
```

et constatez que maintenant on peut aller pratiquement aussi loin qu'on veut. Pour résoudre cet exercice vous aurez besoin de lire posément la documentation de cette classe.

D. Aurions-nous pu donner le même nom aux fonctions **`factorielle1`** et **`factorielle2`** ?

1.6. Quel jour sommes-nous ? Quelle heure est-il ?

But de cet exercice : essayer diverses manières d'obtenir et afficher la date courante (encore un prétexte pour vous faire chercher dans la documentation...).

A. Première manière : la méthode **`java.lang.System.currentTimeMillis()`** donne la date courante, exprimée comme le nombre de millisecondes qui se sont écoulées depuis le 1er janvier 1970 à 0 heures GMT. C'est précis, mais pas très pratique pour organiser sa semaine ! (Retenez quand même l'existence de cette méthode, car elle est bien utile pour mesurer et comparer les performances des programmes).

Écrivez un programme qui affiche le nombre de secondes écoulées depuis le 1er janvier 1970. Exécutez deux fois ce programme, à une minute d'intervalle, et voyez si les valeurs obtenues correspondent à peu près à l'explication.

B. Deuxième manière : ajoutez au programme précédent des instructions qui créent un objet de type **`java.util.Date`** (par une expression comme « **`Date d = new Date();`** ») et le donnent à afficher à **`System.out`**. Comme vous le voyez, c'est mieux, car on obtient bien une date, mais écrite à la manière des anglo-saxophones.

Ne cherchez pas la correction de ce défaut parmi les nombreuses méthodes de la classe **`Date`**, elles sont presque toutes *deprecated* (obsolètes, désapprouvées) et ne méritent pas qu'on s'y investisse.

C. Troisième manière : créer un objet de type **`java.util.Calendar`** (par une expression comme « **`Calendar c = Calendar.getInstance();`** ») et obtenir séparément les éléments de la date (le jour de la semaine, le jour du mois, le mois, l'année) pour les afficher comme bon nous semble.

En étudiant la documentation de la classe **`Calendar`** vous découvrirez qu'on obtient les divers composants par des expressions de la forme « **`c.get(Calendar.MONTH);`** », « **`c.get(Calendar.YEAR);`** », etc.

Des tableaux de chaînes déclarés comme ceci peuvent vous aider à obtenir une présentation adéquate :

```
String[] mois = { "janvier", "février", ... "décembre" };
String[] jSem = { "lundi", "mardi", ... "dimanche" };
```

D. Quatrième manière (la plus « pro ») : construire un objet **`d`** de type **`java.util.Date`** comme dans la deuxième manière et un objet **`f`** de type **`java.text.SimpleDateFormat`** (qui est une variété de **`java.text.DateFormat`**) et afficher le résultat du formatage du premier par le second, par une expression comme **`f.format(d)`**.

Ne vous laissez pas effrayer par la documentation de **`SimpleDateFormat`**. Si vous ne voyez pas comment cela marche, essayez ceci et vous comprendrez :

```
...
Date d = new Date();
SimpleDateFormat f = new SimpleDateFormat("dd MMMMM yyyy HH:mm");
System.out.println("maintenant: " + f.format(d));
...
```

E. Enfin, Ecrire une classe qui mesure et affiche le temps écoulé entre 2 saisies au clavier.