Feuille de TP n° 1 : Java et mots

Pour programmer en Java, toujours avoir sous la main le site de référence de Sun :
`http ://java.sun.com/j2se/1.4.2/docs/api/`

Exercice 1 : Environnement de travail
Se reporter à la feuille décrivant la procédure pour activer son compte.

Une fois votre compte activé, créez dans votre répertoire principal un répertoire `AF3` dans lequel vous mettrez tout ce qui a trait à ce cours, en particulier les programmes écrits au cours de ce TP.

Exercice 2 : La classe `String`
Le but de cet exercice est d'imiter quelques fonctionnalités de la classe `String`, en guise d'échauffement.

1. Écrire une classe `Chaine` permettant de stocker et manipuler un tableau de caractères.
2. Écrire un constructeur `Chaine(char[] caracteres)` permettant de construire un objet de type `Chaine` à partir d'un tableau.
3. Écrire un constructeur `Chaine(int n, char c)` permettant de construire une chaîne contenant `n` fois le caractère `c`.
4. Écrire une méthode `String toString()` qui crée un objet de type `String` à partir de la chaîne stockée.
5. **Note sur la méthode `toString()`** : lorsqu'on passe un objet de n'importe quel type en paramètre d'une méthode d'affichage (comme `System.out.println()`), celle-ci fait appel à la méthode `toString()` de l'objet pour savoir comment l'afficher. C'est pourquoi il est recommandé d'écrire cette méthode à chaque fois qu'on crée une nouvelle classe (sinon, c'est la méthode `toString()` de la classe `Object` qui est appelée, et la valeur renvoyée n'est pas vraiment utilisable.
6. Écrire une méthode `public static void main(String args[])` permettant de tester les constructeurs. Toutes les méthodes définies par la suite seront testées *via* cette méthode `main()`.
7. Écrire les méthodes suivantes (voir leur spécification pour la classe `String` sur le site de référence de Sun, dont l'adresse est donnée plus haut) :
   - `int length()`
   - `char charAt(int index)`
   - `int indexOf(char c)`
   - `boolean equals(Chaine uneAutreChaine)`
   - `boolean startsWith(Chaine uneAutreChaine)`
   - `Chaine concat(Chaine uneAutreChaine)`

Exercice 3 : Les mots avec des chaînes
On veut spécialiser la classe `Chaine` en une classe plus spécifique représentant des mots.

1. Écrire une classe `Mot` héritant de la classe `Chaine`. Définir dans cette classe les constructeurs `Mot()`, `Mot(int taille, char c)` et `Mot(String s)`, et définir une constante `EPSILON` représentant le mot vide.
2. Écrire une méthode `main()` pour vérifier que les méthodes définies dans la classe `Chaine` peuvent être utilisées sans être redéfinies dans la classe `Mot`.
3. Redéfinir la méthode `Mot concat(Mot m)` pour prendre en compte la constante `EPSILON`.

Exercice 4 : Les mots comme des listes
Il peut être plus pratique pour certaines opérations, de représenter les mots comme des listes de caractères.

1. Écrire une classe `MotListe` permettant de représenter des listes simplement chaînées contenant des données de type `char`.
2. Écrire les constructeurs `MotListe()` et `MotListe(char c, MotListe m)` permettant de construire un mot vide, et un mot ayant pour première lettre `c` et comme lettres suivantes le mot `m`
3. Écrire la méthode `boolean estVide()` qui teste si un mot ne contient aucun élément, et les méthodes `char premier()` et `MotListe suite()` qui renvoient respectivement la première lettre de la liste, et le mot privé de sa première lettre.
4. Écrire les mêmes méthodes que dans les deux exercices précédents, y compris la méthode `toString()`