Paquetage, structure de données de base

Exercice 1 - Evaluateur d'expression (suite)

Le but de cet exercice est de bien comprendre la notation de paquetage.
Reprennez l'ensemble des classes de l'évaluateur d'expression et faites en sorte qu'elles appartiennent au paquetage `fr.umlv.evaluator`.
On supposera que par défaut les sources de l'évaluateur sont stockées dans un répertoire nommé `src`.

1. Créer un répertoire "fr" dans le répertoire "src", puis un répertoire "umlv" à l'intérieur du répertoire "fr" et enfin un répertoire "evaluator" à l'intérieur du répertoire "umlv".
   Déplacer l'ensemble des codes sources (.java) dans le répertoire "fr/umlv/evaluator"
2. Insérer dans chaque fichier, en tant que première ligne la ligne suivante `package fr.umlv.evaluator`
3. Compiler l'ensemble des classes en se plaçant dans le répertoire père de "fr" avec la commande suivante : `javac fr/umlv/evaluator/*.java`
4. Exécuter le "Main" à partir du même répertoire
5. Créer un répertoire `classes` de tel sorte que `src` soit son frère.
   Supprimer les fichier ".class" des sous-répertoire de "src"
   Utiliser l'option "-d" du compilateur pour indiquer au compilateur qu'il faut générer les ".class" dans le répertoire "classes".
   Exécuter la classe "fr.umlv.evaluator.Main"
6. A quoi sert la variable d'environnement `CLASSPATH` ?

Exercice 2 - Directive import

Le but est de voir à quoi sert la directive `import`.
On souhaite que la classe `fr.umlv.evaluator.Main` soit renommée en `fr.umlv.evaluator.test.Main`.

1. Créer le répertoire nécessaire.
2. Déplacer la classe `Main` dans celui-ci et faites en sorte que cela compile.
3. Quelle est la différence entre "import" en Java et "include" en C ?

Exercice 3 - Les listes chaînées

Le but est voir comment coder en Java les structures de données habituelles.
Pour la suite de l'exercice, l'ensemble des classes créées devra être créé dans le paquetage `fr.umlv.datas`.
Nous allons dans un premier temps, créer une liste chaînée de chaîne de caractères.

1. Créer une classe `fr.umlv.datas.Link` correspondant à un maillon de la liste chaînée.
2. Créer une classe `fr.umlv.datas.LinkedLink` qui maintient une référence sur le premier maillon de la liste.
   Cette classe devra définir les méthodes :
   1. `add(String text)` qui ajoute un élément avant le premier élément.
   2. `size()` qui affiche le nombre d'éléments de la liste.

3 `toString()` qui affiche le contenu de la liste.

3 Ecrire dans la classe `fr.umlv.datas.Main`, `main` permettant de tester les différentes méthodes.

4 Implanter `String get(int index)` qui renvoie la `index`ième chaîne de caractère.
Que doit-on faire si l'index est invalide ?
Pourquoi serait-il logique de changer l'implantation de `size` pour que la méthode s'exécute en temps constant ?
Ré-implanter `size`.