Interface, classe abstraite, sous-typage

Exercice 1 - Hiérarchie de types et expression

Le but de cet exercice est de construire un évaluateur d'expressions arithmétiques simples. Ces expressions sont représentées sous forme d'arbre.
On veut un type commun `Expr` représentant des expressions arithmétiques qui peuvent être soit une valeur réelle (de type `Value`) soit une opération d'addition (de type `Add`) qui permet d'effectuee l'addition de deux expression.
On veut de plus être capable d'évaluer de ces expressions en utilisant la méthode `eval()`.

```
Expr value=new Value(7.0);
System.out.println(value.eval()); // affiche 7.0
Expr add=new Add(new Value(3.0),value);
System.out.println(add.eval());   // affiche 10.0
Expr expr=new Add(new Value(2.0),add);
System.out.println(expr.eval());  // affiche 12.0
```

1. Écrire les trois classes `Expr`, `Value` et `Add` et leurs méthodes `double eval()`.
2. Implanter la méthode `toString()` sur les différentes expressions.
3. Discuter sur le fait de transformer la classe `Expr` en classe abstraite ou en interface. Faîtes les changements qui s'imposent.
4. Ajouter les classes `Mult` et `Divide` permettant d'effectuer respectivement la multiplication et la division. Refactoriser le code pour mettre à un seul endroit les codes communs.
5. Ecrire une classe `Main` et faites des tests.

Exercice 2 - 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`.

5 Changer la classe `fr.umlv.datas.LinkedLink` pour une implantation plus générique à base d'`Object`.
Que doit-on changer dans la classe `fr.umlv.datas.Main` ?

6 Implanter la méthode `boolean contains(Object o)` indiquant si un objet est où non contenu dans la liste chaînée.