Liste, interface

Exercice 1 - List, LinkedList et ArrayList

Expliquer ce que fait le code suivant :

```
import java.util.List;
import java.util.ArrayList;
import java.util.LinkedList;
...
  public static void main(String[] args) {
    List list;
    if (args.length==0)
      list=new ArrayList();
    else
      list=new LinkedList();

    for(String s:args)
      list.add(s);
  }
...
```

1 Quel différence y a t'il entre `LinkedList` et `ArrayList` ?

2 A quoi sert l'interface `List` ?

Le code précédent compile avec un `warning`, que doit-on faire pour supprimer celui-ci ?

Ré-écrire le code du main en conséquence

De la même façon, transformer le code suivant :

```
 public static void main(String[] args) {
   List list=new ArrayList();
   Collections.addAll(list,args);

   for(int i=0;i<list.size();i++) {
     String s=(String)list.get(i);
     System.out.printf("%s:%d\n"s,s.length());
   }
 }
```

Quel est l'intérêt des `generics` en Java ?

Exercice 2 - Performance sur les listes

Le but de cet exercice est de tester les différences de performance entre les classes `ArrayList` et `LinkedList` sur différents algorithmes.

1 Nous allons dans un premier temps chronométrer le temps d'un parcours d'une `ArrayList` contenant un million `(1 000 000)` d'entiers en utilisant un `Iterator` (pour le parcours).
Utilisez la méthode `System.nanoTime()` pour effectuer une mesure de temps.

2 Modifier le code pour pouvoir facilement chronométrer le parcours dans le cas d'une `ArrayList` ou d'une `LinkedList`.

3 Dans le but de faire d'autres tests de performance, imaginer un patron de conception (**design pattern**) permettant d'effectuer plusieurs tests sur plusieurs types de `List`.
On appelle patron de conception, une façon d'arranger les objets dans un but précis, ici pour effectuer des tests sur des listes.

4 Refactoriser le code existant en utilisant le patron de conception ainsi défini.

Utiliser le patron de conception pour effectuer les tests suivants sur les deux implémentations de `List` :

- parcours de la liste d'un million d'entiers par un itérateur (dans les deux sens).
- parcours de la liste d''un million d'entiers par un index (dans les deux sens).
- en insérant un milier d'entiers en première position dans une liste vide au départ (comme pour une file).

Exercice 3 - Ensemble et Bag

1 Écrire un programme indiquant quels sont les mots qui se trouvent sur la ligne de commande, on affichera les doublons une unique fois.

```
java Unique toto tutu toto titi tutu
```

Dans un premiers temps, afficher "toto", "titi" et "tutu" dans n'importe quel ordre.
Ensuite, afficher "toto", "tutu" "titi" dans cet ordre, c-a-d l'ordre d'insertion.

2 Écrire un second programme pour qu'il compte le nombre de fois qu'un mot apparait sur sa ligne de commande et affiche chaque mot suivi de son nombre d'occurence.
N.B : penser à faire des fonctions assez générales et si possible utilisables avec autre chose que des `String`.