Les collections

Exercice 1 - Comparaison

Ecrire un programme qui prend les arguments sur la ligne de commande et les affichent :

1. dans un ordre randomizé (différent à chaque fois) ;
2. dans l'ordre lexicographique (ordre du dictionnaire) ;
3. dans l'ordre militaire (on compare d'abord la taille des deux chaines avant d'utiliser l'ordre lexicographique ; par exemple, `tb`, `tau` et `tata` sont dans l'ordre militaire).

Pour les deux dernières questions, trouver au moins deux façons (efficaces) d'effectuer ce qui est demandé.

Exercice 2 - Performance sur les listes

Ecrire un programme qui teste les différences de performance entre les classes `ArrayList` et `LinkedList`.

1. Lors d'un parcours de la liste par un itérateur (dans les deux sens).
2. Lors d'un parcours de la liste par un index (dans les deux sens).
3. Lors de l'ajout en première position de valeurs (comme pour une file).

On utilisera la méthode `System.currentTimeMillis()` pour effectuer une mesure de temps.

Exercice 3 - Trions gaiement des points

On veut stocker des points du plan (pitié ne réinventez pas la roue) triés dans une classe `StockPoint`. La méthode `add` permet d'ajouter un point, la méthode `iterator` renvoie un itérateur sur les points triés suivant leur distance à l'origine.

Exercice 4 - Ecrire un getopt objet

On cherche à écrire une classe qui permet d'effectuer le "parsing" de la ligne de commande suivant des options. Une option est un argument commençant par un tiret ('-').

```
interface Parselet {
  String getOptionName();
  boolean collect(Iterator iterator);
}
```

La méthode `getOptionName()` renvoie le nom de l'option. La méthode `collect` est appelée si l'option est reconnue, avec en argument une `List` non modifiable contenant les arguments de l'option (la liste comprend seulement les arguments de l'option courante). La méthode `collect` doit retourner vrai si l'option a trouvé les bons arguments de l'option, faux sinon.

```
ArgumentParser parser=new ArgumentParser();
parser.addParselet(new Parselet() {
  public String getOptionName() {
    return "out";
  }
  public boolean parse(List l) {
```

```
        boolean ok=!l.isEmpty();
        if (ok)
          System.out.println("fichier "+l.get(0));
        return ok;
      }
    });

    parser.parse(args);
```

La ligne de commande

```
      java ArgumentParser -out fichier.txt
```

doit afficher "`fichier fichier.txt`".

1 Programmer la classe `ArgumentParser`.

2 Option pour ceux qui vont vite : ajouter une option par défaut qui permet de collecter les arguments qui ne sont pas associé à une option et une option "-help" qui affiche de façon triée toute les options possibles.

Exercice 5 - Implantons gaiement des sacs

Programmer les `Bag`. Les méthodes à écrire en plus de celles de l'interface à implémenter sont `add(Object o, int n)`, `remove(Object o, int n)` et `count(Object o)`.