# Comparaisons, collections et classes anonymes

## Exercice 1 - Ensemble et Bag

1 Écrire un programme qui indique si un même mot se trouve plusieurs fois sur la ligne de commande.
2 Modifier le programme précédent pour qu'il compte le nombre de fois qu'un mot apparait sur sa ligne de commande.

## 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 - Comparaison

Ecrire un programme qui prend les arguments sur la ligne de commande et les affiche :
1 Dans un ordre aléatoire (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).
Une fois les trois affichages implantés. On souhaite modifier le programme pour qu'il prenne en premier argument une chaine de caractère identifiant l'affichage à choisir.

```
java Order aleatoire titi toto tutu
```

Ici, on demande l'affichage en ordre aléatoire (**aleatoire**). Les deux autres chaînes possibles sont **lexicographique** et **militaire**.
PS: éviter d'utiliser un `switch` pour implanter le choix de l'affichage.

## Exercice 4 - Collection de coordonnées

Ecrire plusieurs implantations de `PointContainer` qui stockent une collection de `Point` triées.
La comparaison entre deux points s'effectue soit :
- en comparant d'abord les abscisses puis les ordonnées ;
- en comparant les points par rapport à leur distance à l'origine.

Les implantations de `PointContainer` devront posséder les méthodes suivantes :
1 Un constructeur prenant en paramètre un ordre de comparaison.
2 Une méthode `add(Point p)` permettant d'ajouter un point.
3 Une méthode `Collection sortedCollection()` renvoyant une collection des points triée suivant l'ordre de comparaison.

Discuter des différentes implantations possibles, et les implanter.
On souhaite maintenant ajouter deux méthodes `Collection xCoordinates()` renvoyant une collection des abscisses des points stockés et `Collection yCoordinates()`.
Implanter ces deux collections sous forme de vues pour l'ensemble des implantations de `PointContainer` existantes.