TP8

Interface

- Écrivez une interface `Ajoutable<T>` contenant une méthode `T ajoute(T x)`. Cette méthode, lorsqu'elle sera implémentée, correspondra à la notion de `+`. Par exemple si on a une classe Entier implémentant Ajoutable<Entier>, cinq.ajoute(trois) renverrait huit.
- Écrivez une classe Entier implémentant Ajoutable.
- Écrivez une classe `Ajoutables` (avec un *s*) contenant une méthode `somme` prenant en entrée une liste d'Ajoutable et renvoyant la somme de ces Ajoutables. Pour cela, le profil de somme sera : `public static <T extends Ajoutable<T>> T somme(List<T> liste)`.
- Utilisez la méthode précédente pour faire la somme des éléments d'un tableau d'Entiers.
- Utilisez la méthode précédente pour pouvoir faire la concaténation de chaines de caractères (pour cela, définissez une classe Chaine implémentant ajoutable)

Élément majoritaire

- Écrivez une méthode statique `elementMajoritaire` prenant en entrée une liste d'entiers et renvoyant l'élément le plus représenté dans la liste. *Indice:* Utilisez une Map.
- Testez votre méthode sur des listes de taille d'un million d'éléments.
- Modifiez votre code pour qu'il fonctionne sur n'importe quel type de liste.

Indice égal à la valeur

Écrivez une méthode statique `indiceEgalElement` prenant en entrée une liste triée d'entiers, `T`, et renvoyant vrai si et seulement si il existe un indice `i` tel que `T[i] == i` . Attention, vous devez écrire ici un code qui fonctionne en temps logarithmique.

Par exemple :

- sur l'entrée `[12,7,9,7]` votre méthode doit renvoyer faux.
- sur l'entrée `[12,1,9,7]` votre méthode doit renvoyer vrai.

Plus proche

Écrivez une méthode statique `plusProche` prenant en entrée une liste triée de doubles, `T`, et un double `x` et renvoyant l'élément de T le plus proche de `x`. Attention, vous devez écrire ici un code qui fonctionne en temps logarithmique.

Par exemple sur l'entrée `[1.3,2.5,7.3], 1.5` votre méthode doit renvoyer `1.3`.