3. Feuille de TP semaine 9 2022_2023

semaine 09 : TP

Personne

Écrivez une classe Personne contenant un attribut nom et un attribut age.

1. Comment trier une List<Personne> par âge croissant ? Écrivez le code ad hoc.
2. Écrivez une méthode qui prend en entrée une List<Personne> et renvoie l'écart d'âge minimum entre deux personnes.

Complexes

On rappelle qu'un complexe s'écrit sous la forme *partie réelle* + *i partie imaginaire* On considère la classe suivante

```
public  class Complexe {
   private double reel;
   private double imaginaire;
   public Complexe(double reel, double imaginaire)
   {
       this.reel = reel;
       this.imaginaire = imaginaire;
   }
 }
```

1. Écrivez une classe implémentant l'interface **Comparator** et permettant de trier une List<Complexe> par partie réelle croissante.
2. Testez votre code dans un exécutable.
3. Comment trier une List<Complexe> par norme croissante ? (la norme d'un complexe est donnée par $reel^2 + imaginaire^2$)
4. Comment trier une List<Complexe> par norme décroissante ?
5. Étant donné une List<Complexe>, comment trouver celui de plus grande norme ?
6. Écrivez une classe **BibComplexes** contenant une méthode prenant en argument une List<Complexe>, un Comparator et un Complexe et renvoyant vrai si ce Complexe est plus petit que tous les complexes de la liste et faux sinon. Testez votre code avec les différents comparateurs écrits précédemment.
7. Ajoutez une méthode prenant en entrée une List<Complexe>, un Comparator de complexes et deux complexes et renvoyant vrai si tous les complexes de la liste sont compris entre ces deux complexes. Testez votre code avec les différents comparateurs écrits précédemment.

Bibliothèque d'entiers

1. Définir une bibliothèque possédant une méthode **occurences** retournant la liste des occurrences d'une liste d'entiers.
2. Ajouter une méthode prenant en paramètres deux listes d'entiers et retournant un booléen indiquant si elles ont les mêmes éléments.
3. Ajouter une méthode prenant en paramètre deux listes d'entiers triés et retournant une liste contenant la fusion (triée) de ces deux listes, sans utiliser la méthode *sort()*.

Algorithmes

1. Deux éléments de somme nulle

Dans cet exercice vous devez écrire un code permettant de détecter s'il existe deux éléments dans une liste dont la somme est nulle.

Écrivez le code d'une classe `LibEntiers` qui permet d'exécuter le code suivant :

```
import java.util.List;
import java.util.ArrayList;
public class ExecutableSommeNulle{
    public static void main(String[] args){
        List<Integer> liste;
        liste = new ArrayList<>();
        liste.add(-10);
        liste.add(11);
        System.out.println(LibEntiers.existeSommeNulle(liste)); // false
        liste.add(10);
        System.out.println(LibEntiers.existeSommeNulle(liste)); // true
        liste.add(7);
        liste = new ArrayList<>();
        for(int i=1; i<200000; i++)
        {
            liste.add(i -1000);
        }
        System.out.println(LibEntiers.existeSommeNulle(liste)); // true
    }
}
```

2. Plus petite différence.

Étant donnée une liste de nombres entiers, on veut trouver la valeur de la plus petite différence positive entre deux éléments de la liste. Par exemple dans la liste `[1, 7, 10, 8]`, la plus petite différence est 1, obtenue en faisant la différence entre 8 et 7.

Complétez la classe `LibEntiers` pour pouvoir exécuter le code suivant :

```
import java.util.List;
import java.util.ArrayList;
public class ExecutablePlusPetiteDifference{
    public static void main(String[] args){
        List<Integer> liste;
        liste = new ArrayList<>();
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // null
        liste.add(-10);
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // null
        liste.add(10);
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // 20
        liste.add(15);
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // 5
        liste.add(7);
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // 3
        for(int i = 0; i < 20000; ++i) {
            liste.add(i - 1000);
        }
        System.out.println(LibEntiers.plusPetiteDifference(liste)); // ?
    }

}
```

Attention : votre code sera testé sur de *grandes* listes. Un algorithme quadratique (c'est-à-dire une algorithme dont dont la complexité est en $O(N^2)$) ne sera pas validé !

Zoo

Finalisez l'exercice sur le Zoo du TD9.