Complément du cours N°7 (classes abstraites & interfaces)

Tri des objets :

- Significations très différentes d'un type d'objet à un autre.
- Chaines de caractères : ordre lexicographique
- Objets représentant des fournisseurs d'une entreprise ?
  Les choix sont nombreux : nom de l'entreprise, numéro de SIRET, chiffre d'affaires, etc.

1) Interface Comparable :

- Possède une seule méthode : `int compareTo (Object o) ;`
  Retourne : `-1 (plut petit), 0 (égal), +1(plus grand)`
- Certaines classes Java implémentent cette interface : String, classes Wrappers (Integer, Float, etc.)

  Exemple : trier une liste de points ?

  Rendre les objets comparables :

```
public class PointPlan implements Comprable{
     ……..
     ……..
     int compareTo (Object o){
          // voir slides du cours N°7 pour le code de la méthode
     }
     ……..
}
```

Utiliser la méthode `void sort (Object[])` pour trier les points :

```
Collections.sort(listePoints) ;
```

2) Interface Comparator :

- Définir plusieurs logiques de comparaisons pour un même type d'objets.
- Exemple : OperationBancaire
  Critères : Date, Montant

Trier les **adhérents** d'un **Club** selon : nom, âge

- Rendre les objets Personne Comparables
- Définir les critères de comparaisons :

  1er critère : noms des personnes

```
**************************
class NomComparator implements Comparator{

     public int compare(Object o1, Object o2){
          String NomPersonne1 = ((Personne)o1).getNom();
          String NomPersonne2 = ((Personne)o2).getNom();

          return NomPersonne1.compareTo(NomPersonne2);
          //compareTo de la classe String

     }

}
```

  2ème critère : âges des personnes

```
****************************
class AgeComparator implements Comparator{

     public int compare(Object o1, Object o2){
          int AgePersonne1 = ((Personne)o1).getAge();
          int AgePersonne2 = ((Personne)o1).getAge();

          if(AgePersonne1 > AgePersonne2) return 1;
          else if(AgePersonne1 < AgePersonne2) return -1;
               else return 0;

     }

}
```

- Utiliser la méthode `sort(Object[], Comparator c)` de Collections :

1er appel (tri par nom) :

****

```
Collections.sort(ListeAdherents, new NomComparator());
System.out.println("Tri des adhérents par nom ");
for(int i=0; i < ListeAdherents.length; i++){
     System.out.println(ListeAdherents[i]) ;
}
```

2ème appel (tri par âge) :

****

```
Collections.sort(ListeAdherents, new AgeComparator());
System.out.println("Tri des adhérents par âge ");
for(int i=0; i < ListeAdherents.length; i++){
     System.out.println(ListeAdherents[i]) ;
}
```

2.1) Deuxième alternative d'utilisation de l'interface Comparator

sans passer par la définition des 2 classes NomComparator & AgeComparator

Comment ? les classes anonymes :

But : création d'une unique instance d'une classe (héritant d'une autre classe ou implémentant une interface) avec peu de méthodes et de lignes de code.

Principes : *Omettre le nom de la classe

*Donner le code au moment de l'instanciation

Dans la classe Personne :

```
public class Personne {

     ……

     public static Comparator NomComparator = new Comparator () {

          public int compare(Object o1, Object o2){
               String NomPersonne1 = ((Personne)o1).getNom();
               String NomPersonne2 = ((Personne)o2).getNom();

               return NomPersonne1.compareTo(NomPersonne2);
               //compareTo de la classe String

          }

     };


     public static Comparator AgeComparator = new Comparator () {

          public int compare(Object o1, Object o2){
               int AgePersonne1 = ((Personne)o1).getAge();
               int AgePersonne2 = ((Personne)o1).getAge();

               if(AgePersonne1 > AgePersonne2) return 1;
               else if(AgePersonne1 < AgePersonne2) return -1;
                    else return 0;

          }

     };

……

}
```

Puis lors de l'appel de `sort()` :

```
Collections.sort(listeAdherents, Personne.NomComparator);
```

Ou (selon le critère choisi)

```
Collections.sort(listeAdherents, Personne.AgeComparator);
```