Ordre

Interface `Comparable<T>` contient une seule méthode :

```
public int compareTo(T o)
```

Cette méthode retourne

- un entier négatif si l'objet est plus petit que l'objet passé en paramètre
- zéro s'ils sont égaux
- un entier positif si l'objet est plus grand que l'objet passé en paramètre.

les classes `String`, `Integer`, `Double`, `Date`, `GregorianCalendar` et beaucoup d'autres implémentent toutes l'interface `Comparable`.

Utilisation

Avec l'implémentation de l'interface `Comparable`, on peut utiliser des méthodes de `Java`qui peuvent automatiquement trier.

- pour stocker des éléments dans un `SortedSet`, il faut que les éléments aient une notion d'ordre !
  ⇨ à chaque fois que l'on ajoute un élément, il est placé au bon endroit pour que les éléments soient stockés du plus petit au plus grand.
- La méthode pour trier une `Collection` se trouve dans `Collections`, elle s'appelle **sort** et c'est une méthode **static** de `Collections`.

```
1 List<Gaulois> villageois = new ArrayList<>();
2 ...
3 Collections.sort(villageois);
```

Après l'exécution de ces lignes, la liste villageois sera ordonnée du plus petit au plus grand en utilisant la notion d'ordre implémentée dans `Gaulois`.

Exemple

```
1  public class Gaulois extends Personnage
2                   implements Comparable<Gaulois>{
3     String nom;
4     int quantiteSangliers;
5       ...
6
7     public int compareTo(Gaulis ixis) {
8       return this.quantiteSangliers - ixis. quantiteSangliers;
9     }
10 }
```

interface `Comparator`

ex : trier les éléments d'une collection : utilisation interface `Collections`

```
   // interface Collections
1  public static <T extends Comparable<? super T»
2                    void sort(List<T> list)
3  public static <T> void sort
4                    (List<T> list, Comparator<? super T> c)
```

```
1  public interface Comparator<T> {
2     int compare(T o1, T o2);
3     boolean equals(Object obj);
4  }
```

Pour comparer des `Gaulois`, et même tous les `Personnage` selon leur taille, on peut écrire la classe suivante :

```
1  public class TriHauteur implements Comparator<Personnage> {
3       public int compare(Personnage gauche, Personnage droit){
4         return gauche.hauteur < droite.hauteur ? -1:
5                (gauche.hauteur == droite.hauteur ? 0 : 1);
5       }
6     }
```

Exemple

Ensuite, on peut utiliser cette nouvelle classe pour trier des `Personnages` selon leur taille.

```
1  public static void main(String[] args){
2    Personnage obelix = new IrreductibleGaulois("Obelix", 1.81);
3    Gaulois asterix = new IrreductibleGaulois("Astérix", 1.60);
4    Personnage cesar = new Personnage("César", 1.75);
5
6    ArrayList<Personnage> personnages =
7                    new ArrayList<Personnage>();
8    personnages.add(asterix);
9    personnages.add(obelix);
10   personnages.add(cesar);
11
12   for (Personnage p: personnages)
13     System.out.println(p.presentation());
14
15   Comparator<Personnage> hauteur = new TriHauteur();
16   Collections.sort(personnages, hauteur);
17
18   for (Personnage p: personnages)
19     System.out.println(p.presentation());
```