TRIER VOS OBJETS JAVA AVEC COMPARABLE ET COMPARATOR

Philémon Globléhi
Développeur Back-end Java

1. Comparable

L'interface Comparable définit l'ordre naturel d'une classe. On implémente compareTo() pour comparer l'objet courant avec un autre.
Exemple : Trier des personnes par âge

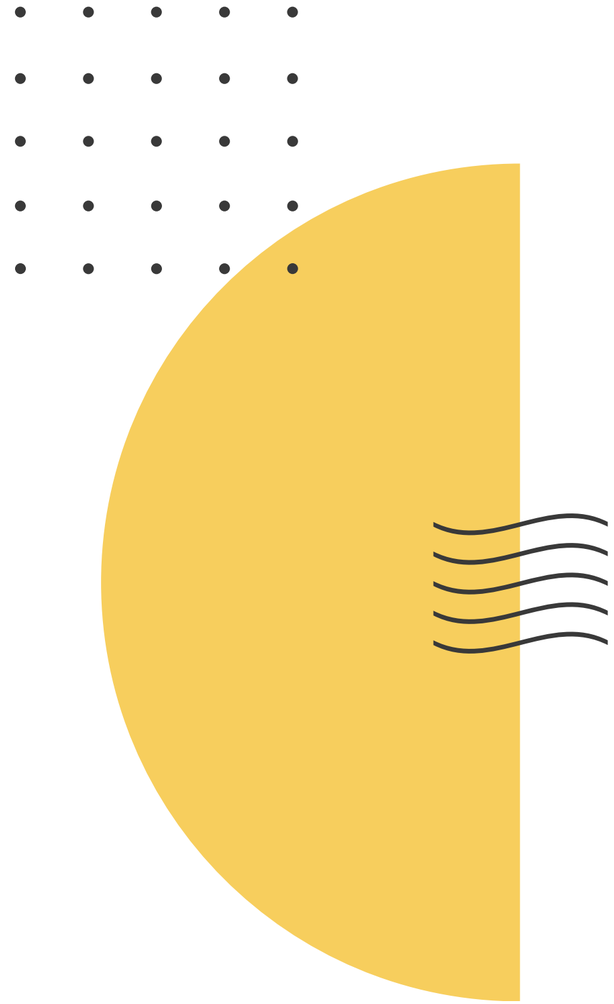
Java

```java
public class Personne implements Comparable<Personne> {
    private String nom;
    private int âge;

    // Constructeur, getters...

    @Override
    public int compareTo(Personne autre) {
        return this.âge - autre.âge; // Trie par âge croissant
    }
}

// Utilisation :
List<Personne> personnes = new ArrayList<>();
Collections.sort(personnes); // Tri automatique via compareTo()
```



À retenir :

- La classe modifiée implémente Comparable.
- Une seule logique de tri possible (ex : âge).

2. Comparator

L'interface Comparator permet de créer des tris sans modifier la classe. On utilise compare() pour définir un ordre spécifique.

Exemple 1 : Trier par nom

```java
public class ComparatorParNom implements Comparator<Personne> {
    @Override
    public int compare(Personne p1, Personne p2) {
        return p1.getNom().compareTo(p2.getNom()); // Trie par ordre alphabétique
    }
}

// Utilisation :
Collections.sort(personnes, new ComparatorParNom());

```

Exemple 2 : Avec une lambda (Java 8+)

```java
// Tri par âge décroissant :
Collections.sort(personnes, (p1, p2) -> p2.getÂge() - p1.getÂge());

```

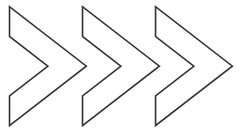
À retenir :

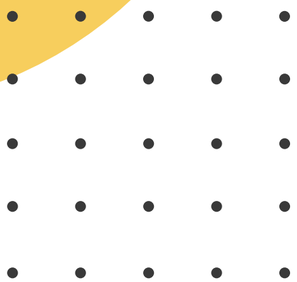
- Flexibilité : Plusieurs Comparator pour une même classe.
- Non intrusif : Aucune modification de la classe cible.

Le récap’

Quand utiliser l’un ou l’autre ?

- Comparable : Pour un ordre par défaut (ex : dates, nombres).
- Comparator : Pour des tris dynamiques ou multiples (ex : tri par nom, puis par ville).





Merci

Philémon Globléhi, Back-end Developer & DevOps Lover
Linkedin: https://www.linkedin.com/in/philemon-globlehi/