Creating Comparators Using Method References

José Paumard

PhD, Java Champion, JavaOne Rockstar

@JosePaumard | https://github.com/JosePaumard

Leveraging method references, default methods and static methods to create comparators

Module Overview

How is a comparator build?

Extracting the information to compare

Chaining Comparators

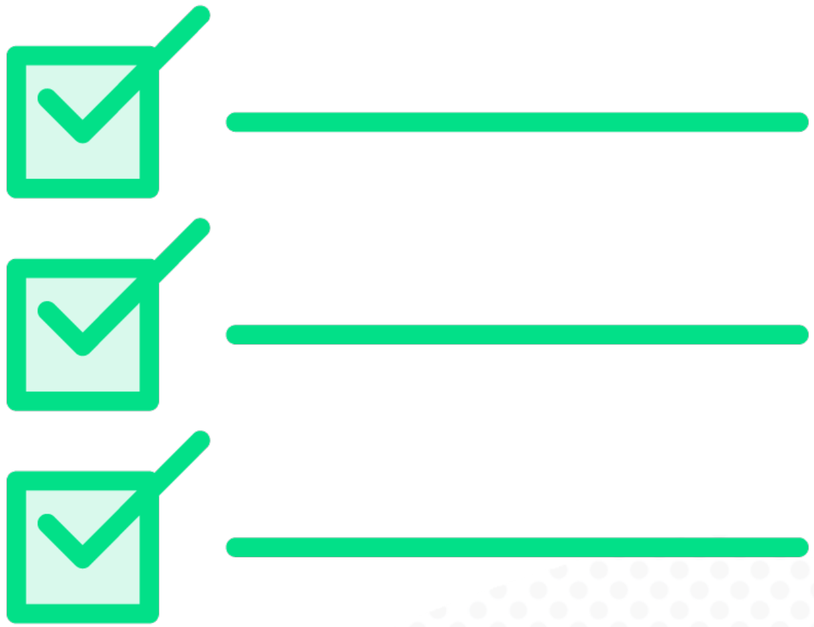
Handling errors and corner cases

What Does a Comparator Depend On?

Two ways to compare objects in Java

Implement Comparable

Or use a Comparator



```java
public interface Comparable<T> {
  public int compareTo(T o);
}

var s1 = "abc";
var s2 = "abc";
var s2 = "def";

System.out.println(s1.compareTo(s2));
> 0

System.out.println(s2.compareTo(s3));
> -3

System.out.println(s3.compareTo(s2));
> 3
```

```
public interface Comparable<T> {
  public int compareTo(T o);
}

record User(String name)
  implements Comparable<T> {
    public int compareTo(User other) {
      return this.name.compareTo(other.name);
    }
}
```

```java
public interface Comparable<T> {
  public int compareTo(T o);
}

record User(String name)
  implements Comparable<T> {
    public int compareTo(User other) {
      return this.name.compareTo(other.name);
    }
}

var users = Arrays.asList(user1, user2, ...);
Collections.sort(users);
```

```java
public interface Comparator<T> {
  public int compare(T o1, T o2);
}
```

```
public interface Comparator<T> {
  public int compare(T o1, T o2);
}

Comparator<String> compLength =
  (s1, s2) ->
    Integer.compare(
      s1.length(), s2.length());
```

```java
public interface Comparator<T> {
  public int compare(T o1, T o2);
}

Comparator<String> compLength =
  (s1, s2) ->
    Integer.compare(
      s1.length(), s2.length());

var s1 = "abc";
var s2 = "def";
var s2 = "ddeeff";

System.out.println(s1.compareTo(s2));
> 0

System.out.println(s2.compareTo(s3));
> -1

System.out.println(s3.compareTo(s2));
> 1
```

What Does a Comparator Depend On?

Writing a Simple Comparator

Writing a simple comparator

By implementing the Comparator interface

Then by extracting the element used to compare your objects

Chaining Comparators

Chaining Comparators

Chaining several comparators

By using default methods

Adding Useful Comparators

Adding Useful Comparators

What about having a natural comparator?

And a way to reverse a comparator?

Handling Error and Corner Cases

Handling Errors and Corner Cases

Handling errors properly

Handling null objects when sorting lists

Module Summary

How the Comparator API is designed

It leverages:

- default and static methods
- method references

How error and corner cases are handled

To get inspiration to design your own APIs

Up Next:

Desiging a Text File Analyzer with Lambdas