

# The Comparator Interface

---

The Comparator interface is similar to the Comparable interface, and the two can often be confused with each other.

Its declaration and primary abstract method are shown here, in comparison to Comparable.

You'll notice that the method names are different, compare vs. compareTo.

Comparator	Comparable
<pre><b>public interface</b> Comparator&lt;T&gt; {      <b>int</b> compare(T o1, T o2); }</pre>	<pre><b>public interface</b> Comparable&lt;T&gt; {      <b>int</b> compareTo(T o); }</pre>

# The Comparator Interface

---

The compare method takes two arguments vs. one for compareTo, meaning that it will compare the two arguments to one another, and not one object to the instance itself.

We'll review Comparator in code, but in a slightly manufactured way.

It's common practice to include a Comparator as a nested class.

Comparator	Comparable
<pre><b>public interface</b> Comparator&lt;T&gt; {      <b>int</b> compare(T o1, T o2); }</pre>	<pre><b>public interface</b> Comparable&lt;T&gt; {      <b>int</b> compareTo(T o); }</pre>

# Summary of Differences

Comparable	Comparator
<pre><b>int</b> compareTo(T o);</pre> <p>Compares the argument with the current instance.</p> <p>Called from the instance of the class that implements Comparable.</p> <p>Best practice is to have <code>this.compareTo(o) == 0</code> result in <code>this.equals(o)</code> being true.</p> <p><code>Arrays.sort(T[] elements)</code> requires <code>T</code> to implement Comparable.</p>	<pre><b>int</b> compare(T o1, T o2);</pre> <p>Compares two arguments of the same type with each other.</p> <p>Called from an instance of Comparator.</p> <p>Does not require the class itself to implement Comparator, though you could also implement it this way.</p> <p><code>Array.sort(T[] elements, Comparator&lt;T&gt;)</code> does not require <code>T</code> to implement Comparable.</p>