# Collections class

<table>
<thead>
<tr>
<th>Method name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>binarySearch(list, value)</code></td>
<td>returns the index of the given value in a sorted list (&lt; 0 if not found)</td>
</tr>
<tr>
<td><code>copy(listTo, listFrom)</code></td>
<td>copies <code>listFrom</code>'s elements to <code>listTo</code></td>
</tr>
<tr>
<td><code>emptyList()</code>, <code>emptyMap()</code>, <code>emptySet()</code></td>
<td>returns a read-only collection of the given type that has no elements</td>
</tr>
<tr>
<td><code>fill(list, value)</code></td>
<td>sets every element in the list to have the given value</td>
</tr>
<tr>
<td><code>max(collection)</code>, <code>min(collection)</code></td>
<td>returns largest/smallest element</td>
</tr>
<tr>
<td><code>replaceAll(list, old, new)</code></td>
<td>replaces an element value with another</td>
</tr>
<tr>
<td><code>reverse(list)</code></td>
<td>reverses the order of a list's elements</td>
</tr>
<tr>
<td><code>shuffle(list)</code></td>
<td>arranges elements into a random order</td>
</tr>
<tr>
<td><code>sort(list)</code></td>
<td>arranges elements into ascending order</td>
</tr>
</tbody>
</table>
The `compareTo` method (10.2)

- The standard way for a Java class to define a comparison function for its objects is to define a `compareTo` method.
  - Example: in the `String` class, there is a method:
    ```java
    public int compareTo(String other)
    ```
  
- A call of `A.compareTo(B)` will return:
  - a value < 0 if `A` comes "before" `B` in the ordering,
  - a value > 0 if `A` comes "after" `B` in the ordering,
  - 0 if `A` and `B` are considered "equal" in the ordering.
Comparable (10.2)

```java
public interface Comparable<E> {
    public int compareTo(E other);
}
```

- A class can implement the `Comparable` interface to define a natural ordering function for its objects.

- A call to your `compareTo` method should return:
  
  a value < 0 if the `this` object comes "before" other one,
  
a value > 0 if the `this` object comes "after" other one,
  
  0 if the `this` object is considered "equal" to other.
NewsSource source1 = new NewsSource("LocalPaper", 22100, 7.9);
NewsSource source2 = new NewsSource("Roommates", 6, 7.1);
NewsSource source3 = new NewsSource("OnlineBlogs", 22100, 7.3);

System.out.println(source1.compareTo(source2));
System.out.println(source2.compareTo(source2));
System.out.println(source1.compareTo(source3));

• What is the output of this program?
(Let -1 be any negative number and 1 be any positive number)

-1 / 0 / 0
1 / 0 / 0
-1 / 0 / -1
1 / 0 / -1
0 / 0 / -1

// first sort on subscribers in ascending order
// then sort on trust rating in descending order
public int compareTo(NewsSource other) {
    if (other.subscribers != this.subscribers) {
        return this.subscribers - other.subscribers;
    } else {
        return (int) (other.trustRating - this.trustRating);
    }
}