
MAKE GIFS AT GIFSOUP.COM

`Collections` class

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

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:

```
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)

```
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`.


ViralHog

pollev.com/cse143

```
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);
  }
}
```