Lecture 18: Comparable

08/05/22

Upcoming

- Checkpoint 7 due Sunday 8/7 @ 11:59pm
- A5 Resubmission due Wednesday 8/10 @ 11:5pm
- A7 due Thursday 8/11 @ 11:59pm

- *(A8 will be released Monday 8/8, due **Tuesday** 8/16)*

- Regrade requests due today!

`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

`compareTo`

- 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`.compareTo(`**B**`)` will return:

A negative number (value < 0)	*(if **A** < **B**)* if **A** comes before **B** in the ordering
Zero (value == 0)	*(if **A** = **B**)* if **A** and **B** are tied (order doesn't matter)
A positive number (value > 0)	(if **A** > **B**) if **A** comes after **B** in the ordering

`Comparable` interface

```
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.