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

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

    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

A. compareTo(B) will return:
Comparable interface

```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.