Java 8: Setup and Review

Notes on the sorting problems:

- The compare method of Comparator should return a negative number if the first entry is "less" than the second, a positive number if the first entry is "greater" than the second, and 0 if they are the same. See the JavaDoc API for details.
- To print out an array after sorting, do System.out.println(Arrays.asList(yourArray)) The point of this is that if you just print an array directly, you do not see anything useful (just the memory address), but if you print a List, it shows the individual elements separated by commas. So, the above trick is simpler than making a loop to traverse the array and print out the elements.

1. Make a new Java project and create a simple class called HelloWorld that does nothing but print "Hello". Run it. This "exercise" is just to make sure you have Eclipse set up properly.

2. Make a static method called "lastEntry" to which you pass a List and get back the last entry of that list. If you pass it a List of Strings, you should get back a String. If you pass it a List of Circles, you should get back a Circle. E.g.:

```
List<Circle> listOfCircles = ...;
Circle lastCircle = ListUtils.lastEntry(listOfCircles);
```

For simplicity, test it with Strings, but be sure you do not do any typecasts in the code that calls your method.

3. Next, support arrays as well as Lists. That is, you should be able to call either ListUtils.lastEntry(someList) or ListUtils.lastEntry(someArray).
Hint: very easy once you think of a minor trick.

4. Make an array containing a few Strings. Use anonymous inner classes to sort it by:

- length (i.e., shortest to longest)
  (Hint: this exact solution was shown in the lecture)
- reverse length (i.e., longest to shortest)
  (Hint: minor variation of the first bullet)
- alphabetically by the first character only
  (Hint: charAt(0) returns the numeric code for the first character)
- Strings that contain "e" first, everything else second. Put the full comparison code directly inside the body of the compare method.
- Strings that contain "e" first, everything else second. But this time, move the comparison code into a static method. For example, StringUtils.eChecker(s1, s2) will return -1 if s1 is "less" (i.e., it contains "e" but s2 doesn't), 1 if s1 is "greater", and 0 otherwise. Now, sort the array as before, but call the static method from the body of the compare method.
  (Hint: simple if you did the previous bullet. But we will use both approaches later.)