CIS 232 - Lab 5

(50 points)

Due: Upload to Canvas by 11:59p Mon. 9/25

Objectives: More practice implementing generics, comparators, and algorithms.

Tasks: Write a program to the following specs and upload your code as before. You can work with one other student on this lab.

Procedure: You can utilize any classes from the Java API as you see fit, *except* for the sort and any comparators, which you must provide the code for.

Write a Java program that:

1. Has a new (application) class, `L5_232XXxx`, that contains an implementation of the static `sort()` method described on p. 247 but using generics similar to the `binarySearch()` method in the `Arrays` class on p. 247
2. The `sort()` method is generic with a `void` return type and two parameters – an array of any reference type, and a `Comparator` that is implemented for the array's type or any of its superclasses.
3. Implement the `sort()` method using actual code (i.e. not via an API method) that will sort the contents of the array according to the `Comparator` object's logic. The sort used must be Selection Sort (yes there are other fine, fine sorts, I want to see you implement this one.)
4. Write a static method in that same class to display the contents of any object array. The method will take in the array plus an integer parameter corresponding to how many items are displayed on a line. Make sure the method newlines at the end of the last line regardless of how many items are on it.
5. Add a `main()` method to the same class that will:
    a. Generate a random array of `Point` (`java.awt` version) objects
        i. Have x/y values range -100 to 100 (inclusive)
    b. Display the original array, sort it, then display the sorted array.
        i. Run the test twice with two different comparators - one for ascending order, one for descending. (put the comparator classes in this same singular class.)
            - Note – Implement both comparators fully and optimally – don't just make one the negated result of the other.
        ii. Criteria for sorting `Point`s for this lab is by x-value, with y-value as a tiebreaker.
        iii. The standard `toString()` is acceptable for output.
        iv. Separate the output into clearly labeled sections.

Upload a zipped copy of your source code by the deadline.

Name your test class (and your source file) with the pattern above. Use proper style. Sloppy, poorly named/formatted, and/or inefficient code will receive significant deductions. Do your best work.