HW 7: Quicksort

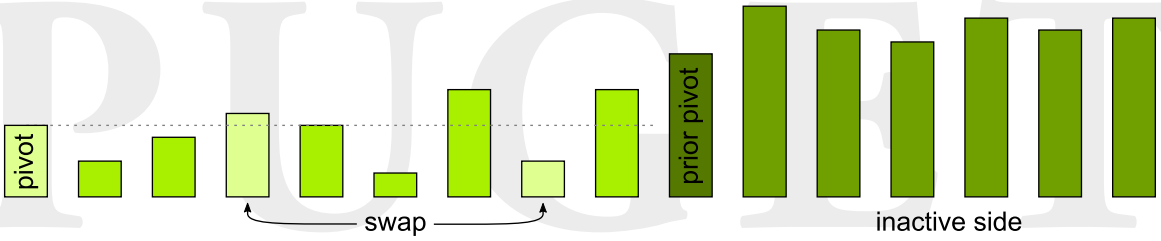
You must implement a class which sorts using quicksort. As before, it should inherit from the abstract Sorter class, and be called `QuickSorter`.

As usual, the `main()` method should ask for a number, and then sort an array of that length, reporting how long (in ms) the sort took:

```
Testing quicksort.
Please enter the array size:  50000
Array of size 50000 took 73 ms to sort.
```

Like mergesort, quick is a recursive algorithm. It consists of these steps:

1. Partition the elements based on a pivot element:
   (a) Pick the pivot (usually the first element).
   (b) Scan forwards from the beginning and backwards from the end, swapping so that elements toward the beginning are less than the pivot and elements toward the end are greater.
   (c) Swap the pivot into the position just prior to where the scanning indices met.
2. Recurse on the elements before the pivot.
3. Recurse on the elements after the pivot.



The ideal pivot would be the median element, that divides the array into two equal parts. Assuming that you find such a pivot, the algorithm runs in linearithmic time, using logarithmic space (for its recursion). Each partition operation is linear, and you partition successively smaller spaces as the algorithm recurses. The average case is much the same.

However, if you consistently choose a bad pivot, the algorithm becomes a quadratic one that takes up linear space. This is because the number of recursive levels becomes linear rather than logarithmic. In fact, quadratic sorts will often beat quicksort in this case! Strangely, if you always pick the first element as the pivot, this means that the pivot will be suboptimal when the array is already sorted. Thus, although quicksort is usually a very fast $O(n \log n)$ sort, it is at its worst when most other algorithms are at their best.

Quicksort's other fault is that it is not stable. These are the reasons that mergesort is used more often for a general-purpose sort. However, quicksort can be a valuable tool, in the right circumstances.

Because quicksort is recursive, you'll need to do the same trick you did before, with both a public and a private version of the sort. In this case, the private version only needs three arguments.

As before, take care that the end user sees no exceptions. Further, you should suppress compiler warnings that are printed during compilation.