HW 5: Heapsort

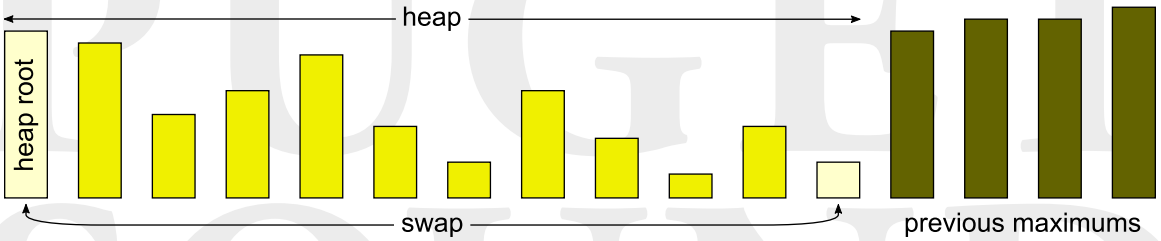
You must implement a class that performs heapsort, inheriting from the provided abstract `Sorter` class. Your class should be called `HeapSorter`.

The `main()` method can be copied nearly verbatim from this week's lab. It should ask the user for a number, and then time how long (in ms) it takes to sort an array of that length:

```
Testing heapsort.
Please enter the array size: 50000
Array of size 50000 took 87 ms to sort.
```

As we have gone over in class, heapsort is a two-stage algorithm:

1. **Make the array into a heap (*"heapifying"*), by performing a *"sink"* operation on all elements with children.** This must be in reverse order, starting with the last element with children and ending with the root, element 0. (A sink operation compares an element with its children. If either child is larger, the element is swapped with the larger child. It must then be compared with its new children, moving downward until either it has no children or it is larger than both of them.) Once every element with children has been sunk, the array is guaranteed to be a heap.
2. **Repeatedly swap the root element with the heap's last element, and then reform the heap.** After the swap, the old root must be in the correct place. It is retired, and isn't part of the heap anymore. Further the the new root is probably too small, and must be sunk to its correct place. The process of swapping and sinking the root repeats until the heap is of size 1, and the array is guaranteed to be sorted.



Both stages can be easily proven to be linearithmic or better (that is, $O(n \log n)$ procedures). The first stage is really linear, though the proof is more complicated. Regardless, the whole sort is dominated by the linearithmic second stage, and so the full sort remains linearithmic.

Please remember to catch all possible exceptions and to handle them intelligently, rather than allowing the user to see it!