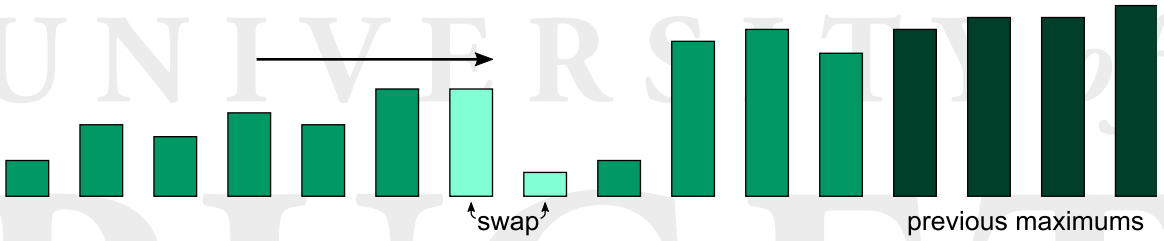
Lab 5: Bubble Sort & Selection Sort

You must implement both bubble sort and selection sort, inheriting from the provided abstract `Sorter` class. Your classes should be called `BubbleSorter` and `SelectionSorter`, and will be in two separate files.

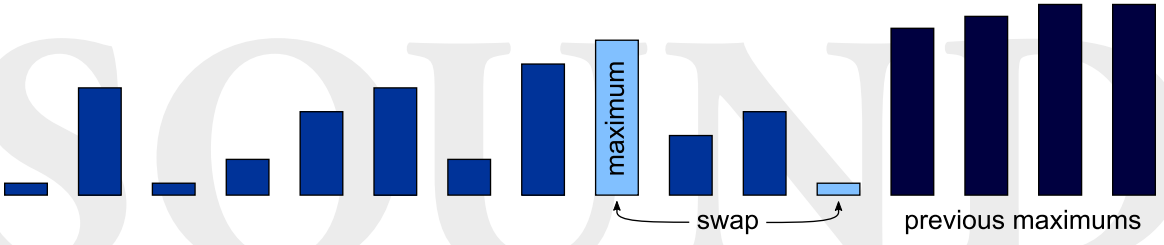
The `main()` method of each class will be almost identical. It will ask the user for a number. It will then use the `timeSort()` method to count how many milliseconds it takes to sort an array of that length, outputting the results. It will **not** output any raw data. For example:

```
Testing bubble sort.
Please enter the array size: 50000
Array of size 50000 took 10854 ms to sort.
```

Recall that bubble sort works by comparing adjacent elements in the array, swapping them if they are out of order. It does this for every adjacent pair from the beginning to the end, so that the maximum element "floats" to the top one step at a time. Then it repeats the whole process, so that the second-greatest element floats up to the second-to-last place. It keeps doing this, so that each subsequent element in turn will float up.



Likewise, selection sort consists of scanning through the array $n - 1$ times. Each time it finds the maximum element and swaps it with the end. (Or, alternatively it can find the minimum element and swap it with the beginning.) Each pass through the array is shorter by 1, so that once an element is swapped to the end, it will never be swapped out.



Note that coding style calls for your program to never print any exceptions to the screen. All exceptions must be caught and handled intelligently, even if the user is being deliberately obtuse (e.g. entering words instead of numbers).