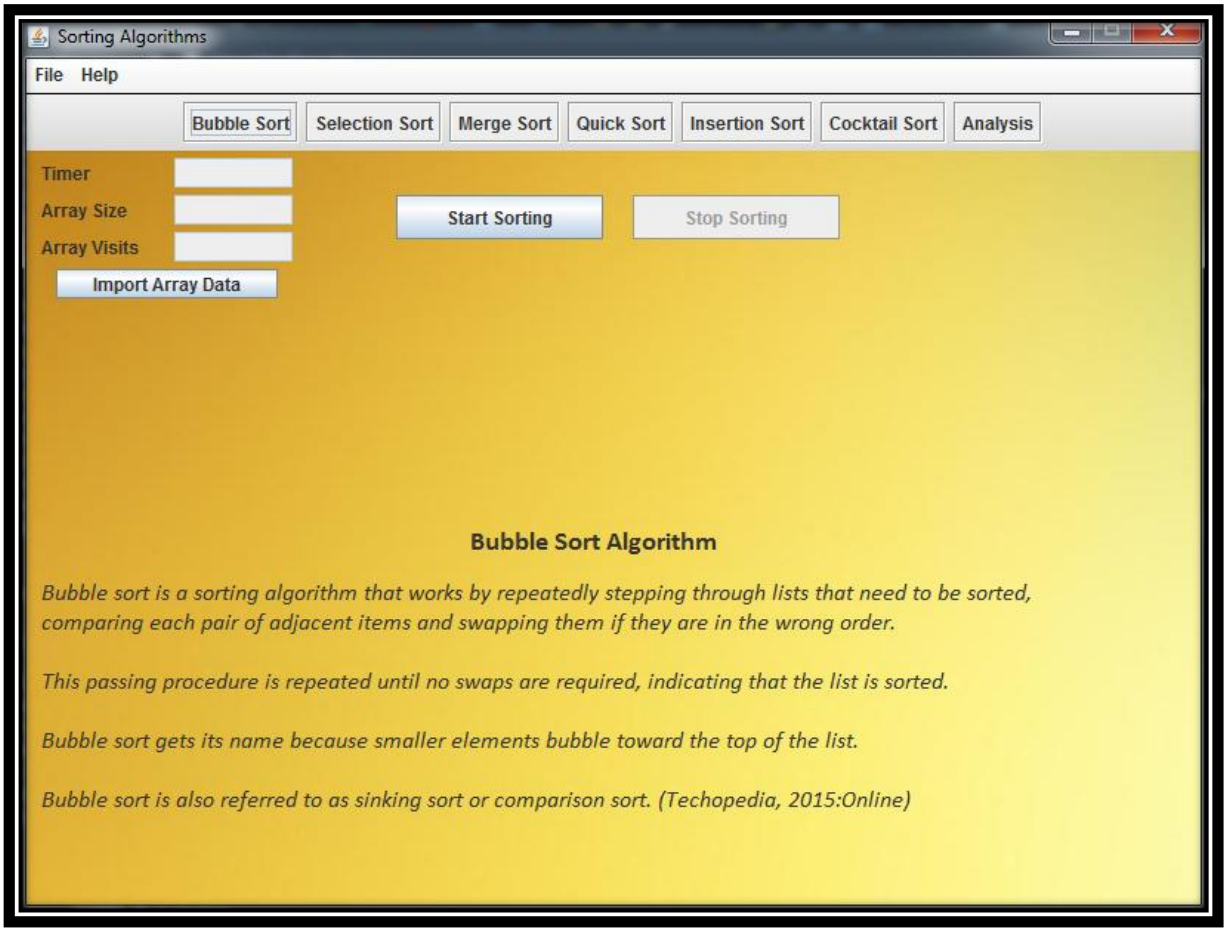
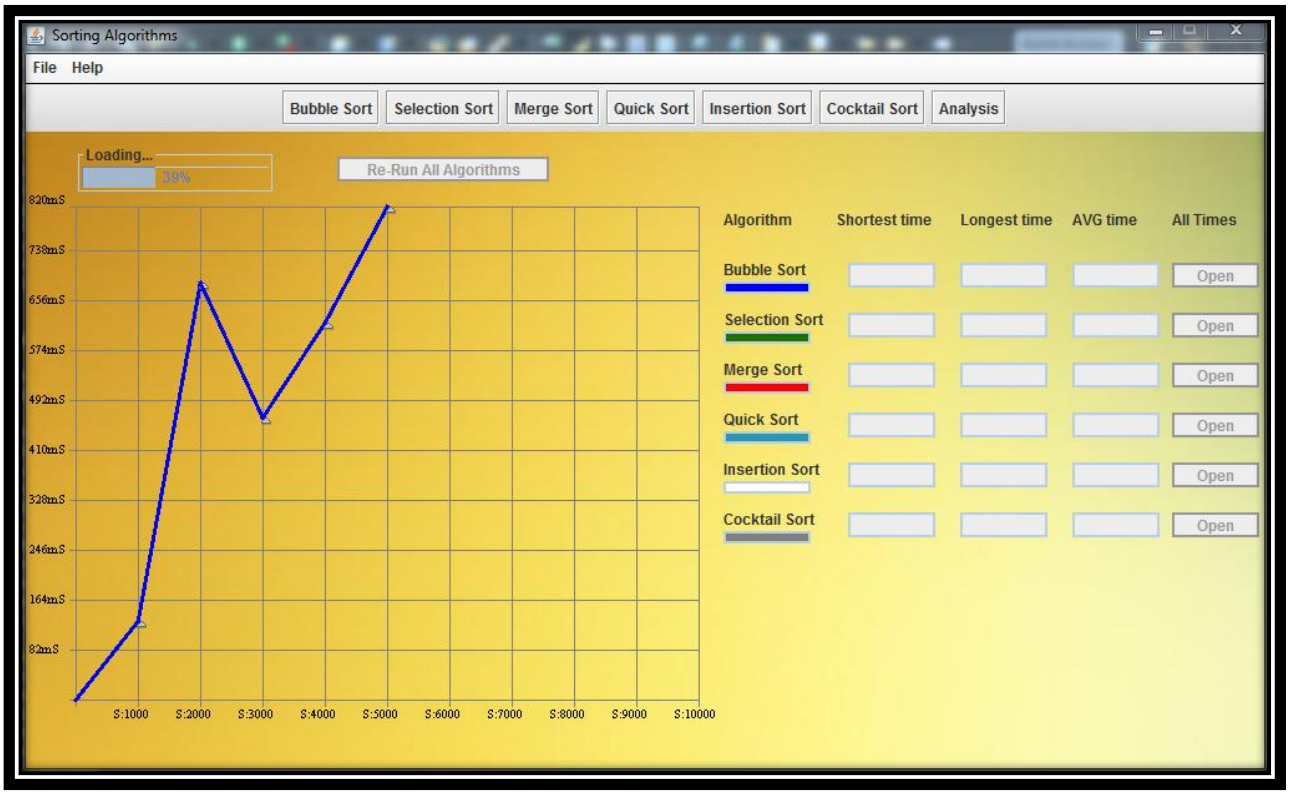
Assignment Report

This snap shot has been taken at the time when the application starts



This Snap shot has been taken when the user clicks the Start Sorting button

This snap shot has been taken when the Analysis button is pressed



This snap shot has been taken once the Analysis thread has completed its sorting tasks

The development of this Java Application made me understand the mechanisms behind various sorting algorithms such as Bubble Sort, Selection Sort, Merge Sort, Quick Sort, Insertion Sort and Cocktail Sort.

- *Bubble sort* moves the biggest elements towards the end of the array, swapping adjacent items until they are all in crescent order. Its complexity is: $O(n^2)$
- *Selection sort* scans through the whole array data to find the smallest value and swap it with the item in the position #1; it then looks for the next smallest element and swap it with the one in the position #2 and so on until it reaches the end of the array. Its complexity is: $O(n^2)$
- *Merge sort* creates sub-arrays of size two, sort and merge them with the following array of size two, creating an array of size four and so on until the data is all sorted correctly. Its complexity is: O(n Log n)
- *Quick sort* divides the array in two parts, moves the lower items on the left side and the higher on the right side after comparing them with a random picked value called pivot; it then re-divides the array in four parts and moves the elements following the same logic. This algorithm keeps dividing and sorting the array until the sub-array has size one, moving from left to the right. Its complexity is: O(n Log n)
- *Insertion sort* scans the array and moves the elements in their correct locations, until all the elements have the right position in the array. Humans use this logic to sort for example a deck of cards. Its complexity is: $O(n^2)$
- *Cocktail sort* uses the bubble sort logic, applying it to both ways so that it scans and swaps from the left to the right and from the right to the left. Its complexity is: $O(n^2)$

The application allows the user to sort random (as default) and fixed data (clicking Import Array Data from any of the sorting methods) and to see the sorting progress on screen along with a brief description of the algorithm.

A timer has been provided on the left hand side along with an array visit counter and the array size. Regarding the Import array data, the user is allowed to import an array of maximum 350 elements; item from the position 351 above will be ignored by the application.

Only one sorting algorithm and the analysis view are permitted to run at the same time and the user can decide to stop and run another algorithm at any time including while the analysis is performing its operations; this thanks to individual threads elaborating the data in the background.

The analysis draws a live graph showing the different running times for each algorithm. It also provides a comparison tool for a better understanding of the sorting methods, including average time, shortest and longest time and a list disclosing all the times an algorithm took to sort array with different sizes.

After running each algorithm and the analysis several times, I have noticed how some of them sort very quickly where others take more time to compute the same task. Quick sort resulted as the fastest one and the bubble sort followed by insertion sort are the slowest ones in my application.

To let the user understand and compare different analysis, the application consents the end user to save the current analysis and load it back in the window whenever it is needed. Another essential tool is the Export as TXT which creates a file containing all the data sorted and unsorted, all the execution times for all the algorithms and all the array sizes.

With this project I have cemented my knowledge regarding:

- The use of separated threads running in background with the class SwingWorker
- Retrieve the data while a different thread is computing it
- Draw graphs according to the data contained in the arrays
- Export data from various arrays to a single file
- Import data from a file and add it to the appropriate arrays
- Create JavaDoc comments and documentation

Some resources have been used to help myself understanding and overcoming difficulties during the life cycle of the application. A full list is included below.

References

Albert Attard (2013) *Swing Worker Example*. [Online][Accessed on 7th January 2015] Available from: http://www.javacreed.com/swing-worker-example/

Chegg (No Date) *Definition of Insertion Sort*. [Online][Accessed on 25th January 2015] Available from: http://www.chegg.com/homework-help/definitions/insertion-sort-3

Data Structures Notes (2009) *What is Merge Sort?* [Online][Accessed on 25th January 2015] Available from: http://datastructuresnotes.blogspot.co.uk/2009/03/what-is-merge-sort.html

Encyclopedia (No Date) *Definition of: QuickSort*. [Online][Accessed on 24th January 2015] Available from: http://www.pcmag.com/encyclopedia/term/55708/quick-sort

Encyclopedia (No Date) *Definition of: SelectionSort*. [Online][Accessed on 24th January 2015] Available from: http://www.pcmag.com/encyclopedia/term/51067/selection-sort

John Purcell (2012) *Multi-threading in Java Swing with SwingWorker*. [Online][Accessed on 8th January 2015] Available from: http://www.javacodegeeks.com/2012/12/multi-threading-in-java-swing-with-swingworker.html

OJ's Perspective (2008*) Sorting Algorithms: The Cocktail Sort*. [Online][Accessed on 27th January 2015] Available from: http://buffered.io/posts/sorting-algorithms-the-cocktail-sort/

Software and Finance (No Date) *Java – Sorting Algorithm – Merge Sort Recursive*. [Online][Accessed on 12th January 2015] Available from: http://www.softwareandfinance.com/Java/MergeSort_Recursive.html

Technopedia (2010) *Bubble Sort*. [Online][Accessed on 25th January 2015] Available from: http://www.techopedia.com/definition/3757/bubble-sort

Vogella (2010) *QuickSort in Java – Tutorial*. [Online][Accessed on 12th January 2015] Available from: http://www.vogella.com/tutorials/JavaAlgorithmsQuicksort/article.html