Detailed System Requirements:

Merge sort, quick sort and new_sort algorithms can be used to sort linear data structures. All of them uses divide and conquer method.

Theese classes written in Java 11 so it can be run in anyplace that java can run. Program starts and sorts 100,1000,10000 length size 1000 arryas for merge sort, quick sort and new_sort then prints the avarage elapsed times for each size and sorting algorithm.

Test Case:

Sorting 100,1000,10000 length sizes 1000 arryas for each sorting algorithm.
Measuring elapsed time for each sorting algorithm and size then dividing the result by 1000 to obtain avarage elapsed time.
Printing the results for comparing the algorithms.

Question 2:

Merge Sort: This sorting approach uses divide and conquer method. There is a MergeSort class for this implementation. It has a public method called Sort which is going to be used to implement merge sort method. Other methods are private
and does the sorting implementation. Sort method direct the array to MergeSort method.
MergeSort method divides the array to be sorted into two half parts called left and right and repeats it recursively until the array parts is going to be one length. All this left and right parts are merged in Merge method.
This method looks first index of the left and right array parts and picks the small value then increases the index of the array part that is used. This process keeps going until the sorted and merged array is get.

Analysis of Time Complexity:

$T(n) = 2*T(n/2) + \Theta(n) \rightarrow \theta(nLogn)$ according to master theorem.

Test Results:

For 100 size array the sorting time is 25 micro seconds.
For 1000 size array the sorting time is 115 micro seconds.
For 10000 size array the sorting time is 1722 micro seconds.

According to the time complexity analysis, when the size of the array increases by 10 times, the elapsed time should increase by 10*log10(base 2) times.

25*10*log10 = 805
empirical: 115
theoretical: 805

115*10*log10 = 3703
empirical: 1722
theoretical: 3703

Results are smaller than expected but close enough (elapsed time is so small so the other factors may effected it) so t

he empirical and the theoretical results are consistent.

Quick Sort: This sorting approach uses divide and conquer method. There is a QuickSort class for this implementation. It has a public method called Sort which is going to be used to implement quick sort method. Other methods are private
and does the sorting implementation. Sort method direct the array to QuickSortRecursive method.
QuickSortRecursive method directs the array to be sorted to QuickSorting method. This method selects the last item of the array as pivot value. Puts smaller values to the left side of the pivot and bigger values to the right
side of the pivot to apply quick sort method. Then returns the index of the pivot. According to this index QuickSortRecursive method calls itself by dividing the array from that pivot index. Which means one call uses array from start index to pivot - 1 and the other call uses array from pivot + 1 to end index. This recursive process keeps going until the length of the arrays going to be one. When this happens the array is going to be sorted.

Analysis of Time Complexity:

$T(n) = T(k) + T(n-k-1) + \theta(n)$

Time complexity depends on the selected pivots. In worst case every time the smallest or the biggest values are selected and because of that the recursive funtion does the recursion n times. In best case every time the middle valued element is selected. So the recursion divides the array to half every time and does the recursion logn times.

Worst Case:

$T(n) = T(0) + T(n-1) + \theta(n)$ -> $T(n) = T(n-1) + \theta(n)$ -> $\theta(n^2)$ according to master theorem.

Best Case:

$T(n) = 2T(n/2) + \theta(n)$ -> $\theta(nLogn)$ according to master theorem.

Test Results:

For 100 size array the sorting time is 6 micro seconds.
For 1000 size array the sorting time is 327 micro seconds.
For 10000 size array the sorting time is 28447 micro seconds.

According to the time complexity analysis, when the size of the array increases by 10 times, the elapsed time should increase by between 10*log10(base 2) and n*n times.

6*10*log10 = 193
6*10*10 = 600
empirical: 327
theoretical: 193-600
193 < 327 < 600

327*10*log10 = 10529
327*10*10 = 32700
empirical: 28447
theoretical: 10529-32700
10529 < 28447 < 32700

Results are between the worst and the best case so the empirical and the theoretical results are consistent

new_sort: This sorting approach uses divide and conquer method. There is a NewSort class for this implementation. It has a public method called Sort which is going to be used to implement quick sort method. Other methods are priv ate
and does the sorting implementation. Sort method directs the array to QuickSortRecursive method.
This sorting algorithm has two recurive part. One of them decreases the array length from both front and end by one and the other one divides the array half. Sort method directs array to new_sort method which directs the array to min_max_finder. min_max_finder finds min and max values of the array and returns their indexes. new_sort meth od swaps min value with the first index of the array and max value with the last index of the array and does recursive call
by decreasing the size of the array by one both from front and end (adjusts index values of the array to do this). At th e end of this process the array will be sorted.

Analysis of Time Complexity:

$T(n) = T(n-2) + 2*T(n/2) \rightarrow T(n) = T(n-2) + \theta(\log n) \rightarrow \theta(n \log n)$ according to master theorem.

Test Results:

For 100 size array the sorting time is 103 micro seconds.
For 1000 size array the sorting time is 5624 micro seconds.
For 10000 size array the sorting time is 487281 micro seconds.

According to the time complexity analysis, when the size of the array increases by 10 times, the elapsed time should increase by 10*log10(base 2) times.

103*10*log10 = 3317
empirical: 5624
theoretical: 3317

5624*10*log10 = 181093

empirical: 487281
theoretical: 181093

Results are bigger than expected but close enough (elapsed time is so small so the other factors may effected it) so th e empirical and the theoretical result are consistent.