Assignment 12
(Full Score: 70 points)

(**Due in class on 4/8/Friday, at Moodle**)

Your name:	Score:

Sorting and Program Performance

Investigate the effect of the array size on the performance, number of comparisons, and number of exchanges for the following search algorithms:

a. Mergesort.
b. Quicksort.
c. Insertion sort.

Requirements:

1. Use the implementations of these three algorithms (static methods *quickSort* (T[] data, int min, int max), *mergeSort* (T[] data, int min, int max), and *insertionSort* (T[] data))provided in the source file of Chapter 8 to test and analyze sorting efficiency.

2. Create a separate main class to perform all required tests.

3. Perform the testing of these 3 sorting methods using different sizes of input data and check whether the code executes according to the theoretical sorting algorithm performance specifications (you need answer this question in your **Lab Report**).

4. Suggested values for the sizes of the test arrays are 10, 100, 500, 1000, 5000, 10000, 50000, 100000, and 1000000 (initialized with random numbers as you did in the previous labs). You may modify these values or add new ones based on the actual running times of your program—you must use at least one array size which will reflect each algorithm's performance specification, that is, this size should be **large**.

5. Record the execution results into the **Lab Report** in the following two formats:
   a. <observed running time> vs. <number of array elements>;
   b. <observed running time> divided by <running time from Big-O analysis of the method> vs. <number of array elements>.

6. Record the analysis of the numbers of comparisons and exchanges into the **Lab Report** for each of the sorting algorithms tested and present the results in the following formats:
    a. <numbers of comparisons> vs. <number of array elements>;
    b. <numbers of comparisons > divided by <number of array elements> vs. <number of array elements>.
    c. <numbers of exchanges> vs. <number of array elements>;
    d. <numbers of exchanges > divided by <number of array elements> vs. <number of array elements>.

7. Use the experimental results from the requirements 3 and 4 above, to justify in your assignment report answers to the following questions in your **Lab Report**:
    a. Is the performance of the code tested consistent with the theoretical big-O analysis?
    b. Which of the specific implementations of the sort algorithms provided in the textbook is more efficient?

<u>Bonus task</u> (**30** points):

Investigate the effect of the initial element ordering on the behavior of the sorting algorithms used in this assignment. In addition to randomly populating the array, test the sorting algorithms with two new arrays (one that is already sorted and the other that is inversely sorted). Present in your **Lab Report** the analysis of your results and the conclusions in the formats described in requirements 5, 6, and 7 above.

Submissions:

- Create a folder (named like “**Assignment12_YourLastName**”) for the project, and copy your source code, **javadoc** file(s), and **Lab Report** to this folder.
- Compress this **folder** including the **Repor**t and **javadoc file** into a single compressed ZIP file and submit it at Moodle by due time.
- (Your code must follow **Java Code/Javadoc Convention** (30% of the credits.))