APCS-A Java Classwork – Number Sorter

Overview

You are to write a menu-driven Java program to demonstrate the use of four sorting algorithms: bubble sort, selection sort, insertion sort, and mergesort. Each sort must be able to arrange an ArrayList of (unlimited) integers in non-decreasing order.

When your program begins, the user should be prompted to enter the name of a disk text file (of integers) from which all of the numbers should be read into an ArrayList. The user should then be presented with a five-option menu that allows the user to choose which of the four sorts (bubble, selection, insertion, merge) should be used to sort the integers into non-decreasing numerical order. The fifth menu option should allow the user to quit the program.

After the user chooses a sort, your program should display the ArrayList of integers in its original (disk text file) order. Then the program should sort the numbers (using the user-chosen sort) and display the ArrayList of integers in sorted order. After that, the program should return to the menu.

Logistics

Your entire program (including a 'main' method) must be contained within a single file named "NumberSorter.java". The disk-reading code, menu, and individual sorts must each have their own method. It is okay if these methods call helper methods (such as the required "swap" method), but do not inappropriately daisy chain methods together. The mergesort requires two methods: one recursive method and one non-recursive method. Your 'main' method should be as short as possible; most of your code should be located in separate methods, with each method performing a single task.

Your program must not contain any global variables. The ArrayList of integers should be defined in your 'main' method and passed as an argument to the other methods. Your program must use an ArrayList of type 'Integer' to store all of the numbers to be sorted. In your individual methods you may use multiple ArrayLists, but you may not use Arrays, HashMaps, or other list-related classes or methods anywhere in your program. Also, you may not use an ArrayList of any type other than 'Integer'.

For reading the integers from the user-supplied disk text file, you may want to reuse code from your previous coding assignment, "Word List Examiner". You will need to modify the code slightly so that it properly reads Integers, not Strings.

Menu and Program Flow

When the five-option menu is shown to the user, it must be displayed neatly, with each option on its own line. Each line should start with a number, and the user will enter a number (1 through 5) to select which sort should be used (or to quit the program).

After a sort is performed, your program must re-display the menu to the user and allow the user to sort the list again (using the same or a different sort) or quit the program. Since the user can perform repeated sorts using the same data set, your program will need to reset the ArrayList to its original (unsorted) state before a new sort is started. Note that if the user chooses to re-sort the list of numbers, your program must actually re-sort the user's list, rather than simply displaying a list in which the integers have already been sorted.

Your method containing the menu should never call itself, and you should not return to the menu by calling it from a method containing sort code. In other words, the method containing the menu must be called from exactly one location. Leaving and returning to the menu method should be handled entirely via 'while' and/or 'do-while' loops.

Swap Method
Your program must contain exactly one "swap" method (named "swap"), which should be called by the bubble sort, the selection sort, and (possibly) the mergesort. The "swap" method must accept three arguments (in this order): the ArrayList of values, followed by two integers representing the indices of the two list values to be swapped.

Error Trapping
Other than what is required by the 'Scanner' class, your program does not need to perform error trapping. You may assume that the user will enter the name of a disk text file that is present and contains only integers. You may also assume that the user will enter only a 1, 2, 3, 4, or 5 when presented with the menu of options.

Reminder
You must always do your own work, and with the exception of the pieces of C++ starter code for the sorts (see below), every bit of code that you submit for this assignment must be entirely written by you, and only by you. Remember that working with another person, having someone else give you code or tell you what to type, or looking at code written by someone other than you is considered cheating, even if you do not directly copy that code into your program.

Starter Code
For this assignment, you are not required to create from scratch, or memorize, the sorts for your program. Instead, you are permitted to use the following online files:

https://redwood.org/demos/cpp/BubbleSort.txt
https://redwood.org/demos/cpp/SelectionSort.txt
https://redwood.org/demos/cpp/InsertionSort.txt
https://redwood.org/demos/cpp/Mergesort.txt

Note that the above files contain C++ code, and they use arrays. You will need to convert them to use Java ArrayLists. Also note that these are the only pieces of external code that you are allowed to use in your program. You may modify the code from the above files in any way that you see fit (as long as the modifications meet the requirements of this assignment), and you are not required to use the above files at all. In other words, you are welcome to write the sorts entirely on your own.

Example Program
On the next pages is the input and output from several runs of my working, acceptable version of this program. In my sample runs I have used the menu options in a way that should provide you with everything you need to know regarding how your program should perform. Your program should provide the user with the same options as my program and should function in the same manner. However, the wording of the menu options and the output from your program do not have to match my program exactly. Note that when I grade your submission, in addition to running your program with my own disk text file, I will also look through your code to make sure your program has been written properly.

Advanced Options
1) Error trap for a missing disk text file, non-integers in the disk text file, and "bad" user-entered data at the menu. You may decide how to respond to bad data.
2) Change your iterative selection sort into a recursive selection sort.
3) Change your iterative insertion sort into a recursive insertion sort.
4) Add a recursive quicksort algorithm and corresponding menu option to your program.
5) Create your own custom sort and add a corresponding menu option. If choosing this advanced option, in your code, right above the extra method that contains your custom sort, use comments to explain how your sort works and how it differs from the other sorts in your program.

```
Program Run #1
==============
DAVE'S INTEGER SORTER PROGRAM

Enter the name of the disk text file containing integers to be sorted:  numbers.txt

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  3

Insertion Sort:
   Original List:  [32, 12, 100]
   Sorted List:    [12, 32, 100]

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  1

Bubble Sort:
   Original List:  [32, 12, 100]
   Sorted List:    [12, 32, 100]

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  5

Goodbye.


Program Run #2
==============
DAVE'S INTEGER SORTER PROGRAM

Enter the name of the disk text file containing integers to be sorted:  list.txt

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  5

Goodbye.
```

```
Program Run #3
==============
DAVE'S INTEGER SORTER PROGRAM

Enter the name of the disk text file containing integers to be sorted:  nums.txt

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  4

Mergesort:
   Original List:  [6, 3, 1, 6, 5, 1, 2, 7, 10, 4]
   Sorted List:    [1, 1, 2, 3, 4, 5, 6, 6, 7, 10]

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  4

Mergesort:
   Original List:  [6, 3, 1, 6, 5, 1, 2, 7, 10, 4]
   Sorted List:    [1, 1, 2, 3, 4, 5, 6, 6, 7, 10]

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  2

Selection Sort:
   Original List:  [6, 3, 1, 6, 5, 1, 2, 7, 10, 4]
   Sorted List:    [1, 1, 2, 3, 4, 5, 6, 6, 7, 10]

Choose which type of sort you would like to use:
   1) Bubble Sort
   2) Selection Sort
   3) Insertion Sort
   4) Mergesort
   5) QUIT PROGRAM

Enter a choice (1-5):  5

Goodbye.
```