The goal of this assignment is provide experience sorting data using selection sort and merge sort.

1. [20] Create a new Eclipse project named Assignment5. Implement a method called createRandomList() which takes no arguments and returns an ArrayList of fifty thousand (50,000) random Strings. The list should not contain any duplicate entries. Hint: take a look at the java.util.UUID randomUUID() and toString() methods.

2. [20] Implement a getSelectionSortedCopy() method which takes an ArrayList as an argument and returns a sorted copy of the list using Selection Sort. Your implementation should not modify the input list. You should implement your own sorting code here, and not simply use built-in Java calls (e.g., in java.util.Arrays). For reference, this process takes ~16 seconds on my machine (with 50,000 entries).

3. [20] Implement a getMergeSortedCopy() method which takes an ArrayList as an argument and returns a sorted copy of the list using Merge Sort. Your implementation should not modify the input list. You should implement your own sorting code here, and not simply use built-in Java calls. For reference, this process takes ~0.25 seconds on my machine (with 50,000 entries).

4. [20] Implement a public boolean isSorted() method which takes an ArrayList as an argument and returns true if the input list is sorted (false otherwise). Use this method to test your sorting implementations.

5. [20] Implement a GUI with at least three buttons:

   1. A button to generate a random ArrayList of Strings using createRandomList().
   2. A button to call getSelectionSortedCopy() by passing the ArrayList of Strings (created by clicking button 1) and report the sorting time on the GUI.
   3. A button to call getMergeSortedCopy() by passing the ArrayList of Strings (created by clicking button 1) and report the sorting time on the GUI.

   Buttons 2 and 3 should be disabled (i.e., not clickable) until button 1 is clicked. Do not reassign the random list to a sorted list when buttons 2 and 3 are clicked, as this will bias subsequent timing runs.

6. [+20] (Extra credit) You’ll notice that the GUI is unresponsive while sorting, which is because each sorting processes is started as a result of an event and is therefore executed on the GUI thread. To avoid this issue, create and execute a new java.lang.Thread for each sorting process.
Zip the Assignment5 folder in your Eclipse workspace directory and upload the .zip file to Blackboard (see Assignment 5 assignment in the Course Documents area).