Programming and Data Structures
Active Learning Activity 10: Sorting Algorithms

Activity Objectives

At the end of this activity, students should be able to:

1. Implement a generic version of the six sorting algorithms seen in class
2. Compare the performance of the sorting algorithms

Activity

---

1. Use the class **`Heap`** from ALA #8.
2. Create the class **`Sort`** with six static methods as described below.
3. Create a static generic method for each of the sorting algorithms with the following signatures:
   a. **Selection Sort –**
   ```
   public static <E extends Comparable<E>> void
   selectionSort(ArrayList<E> list)
   ```
   b. **Insertion Sort –**
   ```
   public static <E extends Comparable<E>> void
   insertionSort(ArrayList<E> list)
   ```
   c. **Bubble Sort –**
   ```
   public static < E extends Comparable<E>> void
   bubbleSort(ArrayList<E> list)
   ```
   d. **Merge Sort –**
   ```
   public static < E extends Comparable<E>> void
   mergeSort(ArrayList<E> list)
   ```
   e. **Quick Sort –**
   ```
   public static < E extends Comparable<E>> void
   quickSort(ArrayList<E> list)
   ```
   f. **Heap Sort –**
   ```
   public static < E extends Comparable<E>> void
   heapSort(ArrayList<E> list)
   ```

For **mergeSort**, define an additional method **subList** to split the list into two halves **firstHalf** and **secondHalf**.

```
public static <E> ArrayList<E> subList(ArrayList<E> list,
                                       int start, int end)
```

The method returns an array list that contains a deep copy of **list** with the elements from index **start** to index **end-1**.

4. In a test program, create an array list of 100,000 integers and fill it with random numbers between 1 and 99,999 inclusive.

5. Sort the array list of random integers using the different sorting methods from class **Sort**. Determine the number of iterations for each sorting algorithm. Do not forget to shuffle the array list after sorting it and before calling the next sorting algorithm. Use the method **shuffle** from **java.util.Collections**.

6. Display the results as a table like the output below and compare the six sorting algorithms based on the actual number of iterations and the time complexity discussed in class.

```
----------  Sample RUN --------------------

Dataset Size: 100,000

Sorting Algorithm       Number of iterations
Selection Sort          5000049999
Insertion Sort          2491809003
Bubble Sort             4999817097
Merge Sort              3537855
Quick Sort              2079757
Heap Sort               1927903
```

7. Submit the Java files **Heap.java**, **Sort.java** and **Test.java** on Github.