Sorting things out!

Lab 11: Quicksort

1. Preparation
    a. Review Section 13.14 Quicksort in the textbook. Step through the animations until you understand how the algorithm works.
    b. (Optional) Explore the quicksort animation using the Visual Sort Comparison Tool at https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html.
    c. (Optional) Step through the two quicksort animations at https://yongdanielliang.github.io/animation/animation.html.

Note: There are many ways to choose the pivot element in the partition step. The author chooses the partition element to be the midpoint of the elements to be partitioned. The two animations both choose the pivot to be the first element or the elements to be partitioned. Partitioning has the same objective in both cases, put all the elements less than or equal to the partition on the left of the partition and those greater than or equal to the pivot on the right side of the partition. Besides how the pivot is chosen, the two animations of partitioning only differ from the method in our textbook in how the completion of the partition is detected. *For this lab, we will use the partitioning algorithm from the textbook.*

2. Project Creation: Create an IntelliJ project for this lab.
    a. Clone your repo for this lab as a new IntelliJ project.
    b. When you have finished execute the main function in SortTest.java. The program should read from a file of integers included in the project and output the numbers with the order unchanged.

3. Complete the *Sorts* class to implement quicksort. The class *Sort*s implements a static method called *quicksort* and a private helper method partition. Complete the methods using the algorithm presented in the text. (Note: We have changed the local variable *l* from the text to *h*. In general it is not good programming practice to use a lower case 'l' as an identifier, as they can be hard to distinguish from the integer value 1.)

4. Test your sort using the *SortTest* class. (Note: SortTest is not a JUnit test.)

5. The sort program reads a file of randomly sorted integers into an array of Integer objects. Your quicksort method should then sort the array in ascending order. The program will print the integers to the console, 10 to the line.
   The first line of the file to be read contains a single integer, which specifies the number of integers that follow in the file. The first line is be followed by the specified number of integers arranged on one of more lines.

6. Modify your sort method so that it counts the number of comparisons. Modify the output of your application program so that it prints the number of comparisons required to sort the file into order. Is your answer what you expected? Why or why not?

7. When your lab is complete, commit it to your repo and push back to your Lab 10 GitHub repo.