HW2 Eric Sabelhaus Solution Documentation

1. Justification of Added Statements to SelectionSort and InsertionSort functions
   1. SelectionSort
      1. I chose to initialize the SelectSort method using an Integer rather than a raw types. This choice was made due to my implementation of the experiment, in which I use the Random class in java.util to create random Integers which are inserted into the array for sorting. By initializing with Integers, it alleviates the need for override statements for the Comparable interface.
      2. I added 8 total statements for iteration and assignment counting, as well as logging. Of the statements,
         1. 2 were integers
         2. 3 were incrementing statements on those integers
         3. 2 were system logging statements for the incrementing integers
         4. 1 was a logging statement to concatenate the overall results of the test
      3. I also cleaned up the layout of
      4. I made no changes to the Swap method utilized by SelectionSort
   2. InsertionSort
      1. I chose to initialize the SelectSort method using an Integer rather than a raw types. This choice was made due to my implementation of the experiment, in which I use the Random class in java.util to create random Integers which are inserted into the array for sorting. By initializing with Integers, it alleviates the need for override statements for the Comparable interface.
      2. I added 8 total statements for iteration and assignment counting, as well as logging of the statements.
         1. 2 were integers
         2. 3 were incrementing statements on those integers
         3. 2 were system logging statements for the incrementing integers
         4. 1 was a logging statement to concatenate the overall results of the test
2. Lessons Learned
   1. Using the data provided from the experiment, I created an extra column on the excel spreadsheet for the assignment deliverables to review how the number of elements modifies the outcome of each algorithm. There is a linear growth in negative efficiency as the size of the index grows when dealing with the SelectionSort algorithm. When dealing with the InsertionSort algorithm, there is a nearly one to one relation of iterations to assignments.
   2. By analyzing the data diagram, it is more visible that the growth with the SelectionSort algorithm is created by the need for more iterations over the data in order to properly sort the array. For every iteration of the outer loop, there must be a full iteration over all elements of the comparable object with the inner loop.
   3. The efficiency of InsertionSort, related to the while statement of its inner loop. The outter loop must always go through a full iteration of all elements, but the while loop does not necessarily need to iterate over all elements of the comparable object. The conditional statement allows for a more efficient selection for assignment, and therefore a more efficient amount of iteration.
3. Data Diagrams
   1. I chose line graphs to display my data.
   2. The X axis represents the length of the array times 100
   3. The line graphs gave a valid depiction of how the data operations between the two algorithms differs.