**Sort of Sort**

Sort of sort method transforms an array of integer into a sort of sorted array. It should be able to handle taking in duplicates, negatives, and any length of array. And in my test class, I create tests where the method could handle different situations. In the first test, I look to see if the general functionality of the method worked. I declare an **int[] array = {n};** and then declare an expected value **int[] expected = {n};** and lastly use the **junit** tests to pass class and methods to the tests. Secondly, I declare the same variables but this time I put different values inside of them, in this case I was testing the method if the array had negative values. On the third test, I declare the same int array variables but now I pass it duplicates to see if the method would sort them properly. On the fourth test, I test the method if the array length was 1. And finally, I test the method by placing negatives, duplicates, and the length of the array is longer than expected. All tests successfully worked.

Best case scenario would be n because in an array on integers, the array of the length would be n and it would just have to go through the array once. A worst case scenario would be if it was a 2D array, then the method would not work because it can only sort numbers in a 1D array.