Jason Greenlaw

CS 1632 – DELIVERABLE 4

PROPERTY-BASED TESTING

http://www.github.com/jasongreenlaw/deliverable-4

**Summary**

For this assignment, I chose to use property-based testing instead of combinatorial testing. I chose this because I wanted to explore what it would be like to test a built-in java method like the Arrays.sort() method. By doing this assignment, I learned more about the intricacies of property-based testing.

The first step in my process of doing property-based tests was briefly looking back over the lectures and learning more about the properties that an array sorting method should have. These properties include the output array having the same length as the input array, the values in the output arrays are always decreasing in value, every element in the input array is in the output, no element not in the input array is in the output array, running the algorithm more than once should not change the output, and running the algorithm twice should always produce the same resulting output array. With this information I narrowed down the test properties to three parts: 1) The output array is always the same size as the input array 2) Every element in the input array is in the output array, and 3) Values in the output array are always increasing or equal. I split these properties into three separate test cases and used the JUnit @Before tag to initialize random length and value arrays before each case.

I did not really run into many problems in this assignment. The hardest part was writing the method to set up the random length and value arrays. I used a two dimensional array to store 100 individual arrays. Then I randomized the length and values with built-in Java random utilities. I allowed the values stored in the array to be anywhere from 0 to 2^31 – 1 since I wanted to test a large range of values to ensure that the properties held even with high or low values.

In the end I was able to learn a lot from this assignment. Testing for properties forced me to think in a slightly different way than I had previously imagined. This allowed me to gain a solid understanding of how sorting methods should work in terms of their properties, not just their inputs and their outputs. I believe that thinking from the perspective of the properties of a sort method allowed me to create more robust JUnit tests that I would have written before.

**Screenshot of JUnit Tests**

