Sam Kopansky

<u>CS 1631 – DELIVERABLE 4</u>
<u>PROPERTY-BASED TESTING</u>

This deliverable is my shot at redemption. I didn't do as well as I'd hoped on Deliverable 2 and since I had the option to try again at JUnit tests I figured I had to do it again, but do it right this time. This assignment isn't exactly the same in the sense that I am not writing unit tests, but using the JUnit libraries and writing asserts that are actually useful is something I want to get good at. Unfortunately, eclipse didn't share my dream and kept giving me errors every time I would run any test program, including the default test program that eclipse creates by itself when you make a new JUnit test file.

Unwilling to give up on testing, I made my own little JUnit program as you suggested and ran my tests as methods in a java program. The program simply sees which tests passed and which didn't. It then uses that information to display the number of passed tests, the number of failed tests, "GREEN!" if all tests passed, and "Red ☹" if at least one failed.

The tests I ran were four different properties of what a sorted array should look like. The first test I made checked that the lowest number in the array was first in the array since the Arrays.sort() function put all values in ascending order. I did this by getting the minimum random value that was generated while populating the array and comparing that to the value at lowest index of the sorted array. The second test I made checked that the maximum value was last in the array. I accomplished this by determining the largest randomly generated value to be put in the array and compared that to the value in the last index of the sorted array. The third test I made checked that every value in the array was lower than the one after it. I did this by going through the sorted array and checking that the value at every index was lower than the value at the index after it. Finally, I checked that sorting an array twice would give the same sorted array back both times. I made this happen by sorting an array twice and checking that the values were the same at each index.

Moving on to what I learned, I'd say that I picked up two main things from this deliverable. The first nugget of wisdom is that I should never trust eclipse and that I never plan on using again unless I am given no other option. The second thing I learned here is that property based testing makes a lot of sense when trying to make an application work the way you expect it to. Unit testing also does this, but I feel like unit testing abstracts in the wrong direction. I'm going to refer to this as negative abstraction. It breaks up a property into such tiny pieces that you would only know if every little chunk of code works, but you may not know if they work together properly, therefore you may not know if the property is actually executed correctly as a whole. Unit testing definitely has it's applications but then trying to figure out if something works the way you want it to in terms of the requirements of the program, property based testing seems like a good strategy to include in your test plan.