Clinton Wadley

CS 1632 – Deliverable 4

Property-based Testing

github.com/cwadley/property-testing

Summary

For deliverable four, we could choose to do either property-based or combinatorial testing. I chose to do property-based because I found this different approach to testing interesting, and I wanted to further explore what properties are good to test, and how one goes about testing them.

As per the instructions, I tested the Java Arrays.sort method. I wrote a JUnit @BeforeClass method, which generates three identical two-dimensional arrays: *controlArrays, sortedArrays,* and *doubleSortedArrays*. Each of these two-dimensional arrays contains 150 randomly sized arrays of integers. It then sorts each array within *sortedArrays*, and sorts the arrays within *doubleSortedArrays* twice. The unsorted *controlArrays* is used as a baseline to compare the other, sorted arrays against.

I then wrote several JUnit tests to test the properties of the arrays. The properties that I chose to test are:

* The length of the sorted arrays should match the unsorted arrays
* The sorted arrays should be in ascending numerical order
* The sorted arrays should be of type int[]
* The sorted arrays should contain all of the same values as the unsorted arrays
* Sorts on an array should be idempotent

The most difficult part of the implementation was figuring out how to test some of the properties. It took a bit of research and experimentation to figure out how to test the type of an array. Other then that, the implementation was quite straightforward.

This exercise showed me how the property-based testing approach has significant advantages: the tests are relatively easy to write and they test a large number of scenarios very quickly. The approach does have its weaknesses, however, in that it is not ideal for testing all aspects of a program. Code not encapsulated within an object, for instance, would be more difficult to test since there is no object to even have properties. It also does not look for specific output, and can therefore not guarantee correctness. Therefore, a blended approach, using both property and traditional unit tests, would likely be a more ideal solution.

Screenshot

