**SlideGame Testing Report**

**JUnit Tests**

“*First, middle, last*” and “*0, 1, many*” are ways to test a program. “*First*” refers to the first part of the tested item. The method should be tested against the first part of the item. “*Middle*” refers to the middle part of the tested item. The method should be tested against the middle part of this item. “*Last*” refers to the last part of the tested item. The method should be tested against the last part of the tested item. “*0*” is to use test cases of values of 0. “*1*” is to use values of 1 and non-zero values around one. “*Many*” is to use values larger than non-zero values around one. If the method is subjected to these two forms of testing, then most test cases have been considered and the method can be thought of as well-checked.

The JUnit tests I have used cover all of these cases. In a more general form of testing, I first set the dimensions of the arrays to different sizes. These sizes will produce weird game boards, but assure the methods work in all cases and sizes of the game. These weirdly dimensioned arrays, and arrays of normal dimensions, are then tested against the “*First, middle, last*” and “*0, 1, many*” forms of testing. Testing the “*first*” part of the 2D array consisted of giving the first row and first column of the array values of zero, non-zero values around one, and larger values. All eight methods were tested successfully against this part of the array. Testing the “*middle*” part of the 2D array consisted of giving the middle parts of the array, any part of the grid that is not a border. The middle of the array was given values of zero, non-zero values around one, and larger values. All eight methods were tested successfully against this part of the array. Testing the “*last*” part of the 2D array consisted of giving the last row and last column of the array values of zero, non-zero values around one, and larger values. All eight methods were tested successfully against this part of the array.

**\*\*The JUnit testing resulted in eight correct methods\*\***

To verify the game is working, I have been playing it for a week now. I have reached 2048 many times, and have not noticed a single occurrence of an unexpected slide from a button click, an incorrect outcome from a slide method, any tile colored incorrectly, the motivational quotes incorrectly displayed, or the reset button not working.

The slide methods were very specific. The most complicated were the cases of {0, 2, 1, 1} and {0, 1, 1, 2}. If the row {0, 1, 1, 2} were to be shifted left, the outcome would become {4, 0, 0, 0} while {0, 2, 1, 1} becomes {2, 2, 0, 0}. These were the trickiest cases of every method, but they are 100% working. I even initiated games of sizes 10 x 3 or 20 x 4 and all cases were working. Rows of even length and each column of same value would condense into twice the value and half the occurrences. The occurrences will be against the right if the slide right method was used and on the left if the slide method was used. The same is true with the slide up and slide down methods, but along the columns. If the next non-zero value in the array is not equal to the current, then the next non-zero value is placed directly after the current value and the current value is then incremented. For example, {2, 0, 0, 1} -> {2, 1, 0, 0}. Otherwise, the value of the current is double and the next non-zero value is set to 0. For example, {2, 0, 0, 2} -> {4, 0, 0, 0}. A combination the examples is {2,0,2,1} -> {4, 0, 0, 1} -> {4, 1, 0, 0}. The methods needed to be tested for the return too. The methods were tested to see if their return matched if they had changed or not.

Once I believed the game to be finished, I handed it off to other students in my dorm for further testing. Each would play one game til they won or lost. Each student reported no problems in the testing, but complained about gameplay. The biggest complaint was surely the lack of sliding animation, but I’ve no clue where to even begin to implement this. Overall, the project was a success.