Testing Report SlideGame

Junit testing for slideLeft

For slideLeft I had 8 cases. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideRight

For slideRight I had 8 cases. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero then two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideUp

For slideUp I had 8 cases. This time I needed to expand the testing array because it would use multiple rows. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideDown

For slideDown I had 8 cases. Again this array is multiple rows like slideUp’s. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideUpLeft

For slideUpLeft I had 8 cases. This along with the rest of the diagonal methods requires an array with multiple rows and columns. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideDownRight

For slideDownRight I had 8 cases. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideUpRight

For slideUpRight I had 8 cases. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Junit testing for slideDownLeft

For slideDownLeft I had 8 cases. My first test was a zero test where it was an array full of only zeros. It returned false as expected. My next test tested for a single number sliding which functioned properly. My third test checked the case provided on blackboard of a zero then four ones. It returned correctly two twos in the correct locations. The next test had a zero than two ones and a two. This test correctly returned a four that was properly slid. The next test checked for many at once, where it had a 1, 0 ,2, and 3. It returned 123 all slid correctly and passed the test. Then next test checked for many where adding would be required. I had the input as a 1, 1, 1, and 2. It returned the correct results of 2, 1, 2. The next test checked if the entire grid was filled with ones and no zeros. It correctly returned 2, 2 and slid them properly. My final test checked if the input was an already slid one if it would return false, which it did.

Testing for the GUI of SlideGame

One of the main problems I had with the GUI was random spawn spawning an incorrect number of interfaces, specifically the diagonal methods. As you can see by trying it that only one random number will spawn now. The GUI allows all buttons on the sides of the grid to be clicked and they execute the appropriate methods. The corners all call diagonal methods, and the middle buttons do nothing. The methods all work appropriately and quickly, and only add when appropriate. Also the main method can take parameters and handle exceptions. One parameter number becomes a square of that dimension, and two becomes a rectangle. If it is blank it constructs a default 4x4 grid, and if a non-number is entered it responds by saying it is invalid input. Also if a number larger than 50 is input it responds with saying the number is too large. One test I performed was to count and make sure only one new 1 was spawning at a time which worked. Another test was watching a specific number and making sure it was adding correctly, for example I would try and add the bottom right number of the board as high as I could. This I tested all the way up to 128, and it should work far beyond that. The GUI will also not spawn a number if the button clicked cannot move anything. In that case nothing will happen until another button is clicked that can operate on the map. The game also no longer crashes at the end when no more moves can be made.

Extra Credit

Well it’s fairly evident but it changes the background color, font color, font size, as well as font itself as the number on the grid increases increases. Additionally I stopped the game from crashing at the end.