Count Positive:

1. The test fails because it counts positive and negative numbers except when 0 is put into the array. 0 is not positive.

Bad

@Test public void arrayContainsZeroes()

{

int arr[] = {-**4, 2,** -**1, 2, 0**}**;**

*assertEquals*("Array contains zeroes"**,2,** CountPositive.*countPositive*(arr))**;**

}

Good

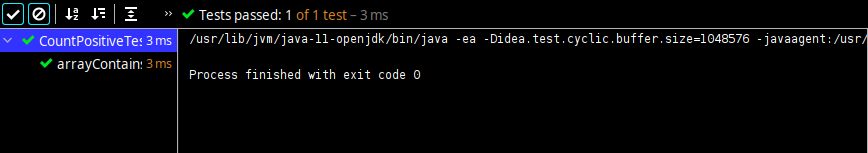
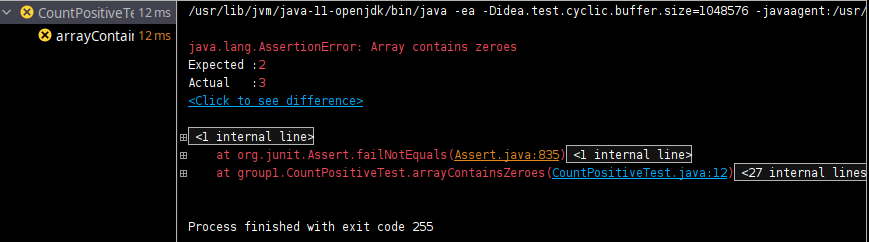
@Test public void arrayContainsZeroes()

{

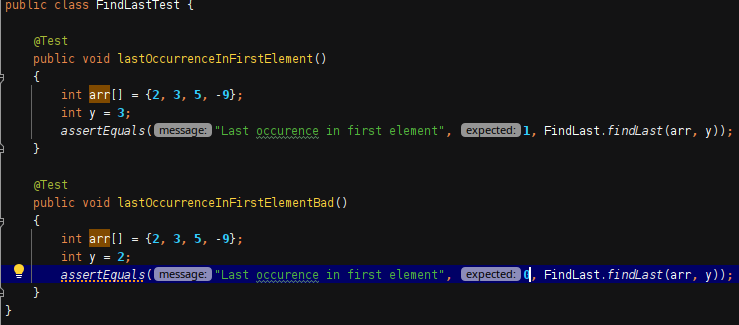
int arr[] = {-**4, 2,** -**1, 0**}**;**

*assertEquals*("Array contains zeroes"**,2,** CountPositive.*countPositive*(arr))**;**

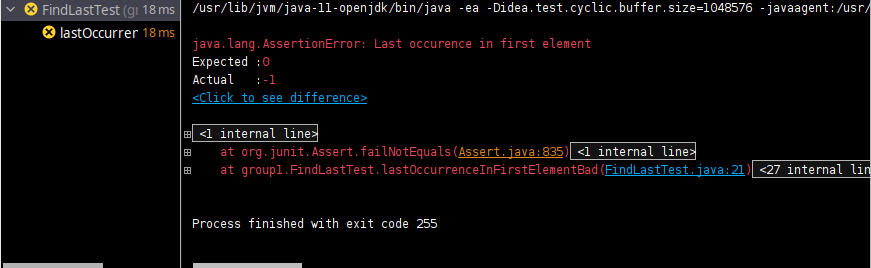
}

3. 

FindLast

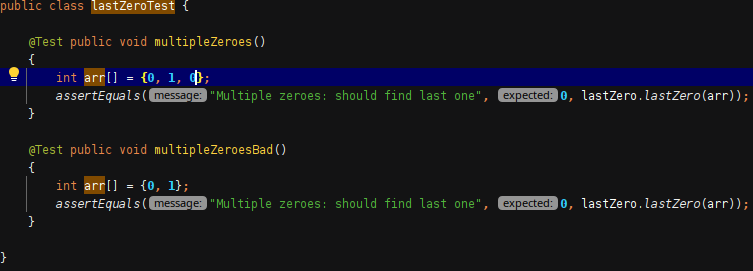
1. The test fails because it finds the index of the last occurance of the input or the highest index. The test fails because it can’t identify its own first number in the array and returns -1 due to that fact.
2. 

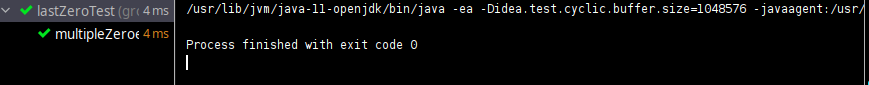
3.



lastZero

1. The test fails because when running through the loop it returns zero immediately because it doesn’t have the chance to return the last element (zero). Thus, when running the test, it should be expected to have the 3rd element or 2nd in the array. There are multiple zeros when the test is run and if only one zero is in the array it still passes even though there aren’t multiple zeros.

1. 

3. 

OddOrPos

1. The test fails when adding additional negative numbers to the array. It doesn’t count any odd negative numbers and the expected value still passes when at the same number.

Bad

@Test public void negativeOddNumbers()

{

int arr[] = {**1, 2, 3,** -**1**}**;**

*assertEquals*("Negative odd numbers in array"**, 4,** OddOrPos.*oddOrPos*(arr))**;**

}

Good

@Test public void negativeOddNumbers()

{

int arr[] = {**1, 2, 3**}**;**

*assertEquals*("Negative odd numbers in array"**, 3,** OddOrPos.*oddOrPos*(arr))**;**

}

3. 