Test Evaluation for code 9455

Here I am going to analyze the code 9455. In this code, I ran the test plan 5061. The result is shown below for each case:

Test Case 1:

Checking if the function **isSorted** can correctly determine a sorted array or not

Input: 6 7 9 13 21 45 101 102 150 (a sorted array with square dimension)

Result: it correctly returns a true value

Function: In this test assertEquals uses to check the equality between a given sorted array and true.

assertEquals(true, TableSorter.isSorted(Table.GetTable("sorted_array.txt")));

Test Case 2:

Checking if the function **isSorted** can correctly determine a not sorted array or not.

Input: 13 7 6 45 21 9 101 102 67 (a not sorted array with square dimension)

Result: it correctly returns the false value

Function: In this test assertEquals uses to check the result of **isSorted** with false.

assertEquals(false, TableSorter.isSorted(Table.GetTable("unsorted_array.txt")));

Test Case 3:

Checking the function's behavior to the non-Int array.

Input: 6.2 7.1 9.5 -13 21.1 45.01 101.0 102.1 150.01 (an array with double digits)

Result: It gets an exception, but the test cannot handle it and cause an error in the test. Fail result. By catching the error, this problem can be solved.

Function: this test uses the assertEquals and compares the return value with false.

assertEquals(false, TableSorter.isSorted(Table.GetTable("nonInt_array.txt")));

Test Case 4:

Checking if the program is robust when the input is a negative array or not.

Input: -602 -107 -92 -23 -21 -15 -11 -10 –5 (negative integer array).

Result: It returns false. That means the program cannot handle negative integer and the test catch this error correctly.

Function: this test uses assertEqualitys and compares the result by false value.

assertEquals(false, TableSorter.isSorted(Table.GetTable("negativeInt_array.txt")));

Test Case 5:

In this test case, the program checks about the positive Integer array.

Input: 6 7 9 13 21 45 101 102 150 (a sorted positive integer array)

Result: It returns true, but the function compares the result with false. This is wrong and it must compare with true. So, the test case fails in this case too.

Test Case 6:

In this test case, the test checks the behavior of the program when the inputs are not in a square number.

Input: 6 7 9 13 21 45 101 102 (not in a square number of elements)

Result: the test throws an error. It would be better to catch the error. But the program can find an array without the square number of the inputs.

Test Case 7:

In this test case, the program checks an input with the square number of the elements.

Input: 6 7 9 13 21 45 101 102 150

Result: the test correctly returns the true value. This means the function is worked with the square number of elements as an input.

Test Case 8:

The objective of this test case is to check function behavior with one element as an input.

> Input: 4
>
> Result: It returns true correctly and that means the function can handle one single element as an input too

Conclusion:

The test works well on all 8 different test cases. In two test cases (3, 6) it would be better to handle the exception in the test function does not throw it out into the console. In test case 5, the value of false must change to true.

The rest of the code works well, and it can check the functionality of TableSorter very well.

The code is documented very well and clear for the user tounderstand. Because I did BB test, so I cannot analyze the code. The only thing I can say, based on the inputs and outputs that I got, program doesn't have any issues or problem.