BB\_Test\_Teams

**Program reviewed** 4680

**Test plan used** 6314

**Test Case 1:**

* + Input:
  + Expected Output: Accept Array

Reason: The input for this should be **a file with a single space** inside of it, which we can interpret as a 0x0 array i.e. an empty array. While this is counterintuitive to the purpose of the lab, an empty array should still be accepted by the program

*The program does not accept files; however, the user can manually modify the values of the array. For this test case, the program returns an error when the array is empty, failing this test case.*

**Test Case 2:**

* + Input:
  + Expected Output: Error
  + Reason: The input for this should be **an empty file**. As opposed to test case 1, this input should NOT be accepted because there is nothing being read from the file.

*The program does not accept files resulting in an error failing this test case.*

**Test Case 3:**

* + Input:
    - 1
  + Expected Output: Accept Array
  + Reason: The input file will be a 1x1 array, a single value which also counts as a square array and thus, should be accepted by the system.

*The program was able to accept a single value (1) returning true passing this test case.*

**Test Case 4:**

* + Input:
    - 1 2 3
    - 4 5 6
  + Expected Output: Error
  + Reason: The input file will be a 2x3 array, which is not a square array and should not be accepted by the system

*The expected output was correct, the program returns the following string “ERROR!: Input must be square” passing this test case.*

**Test Case 5:**

* + Input:
    - 1 2
    - 3 4
    - 5 6
  + Expected Output: Error

Reason: The input file will be a 3x2 array, which like case 4 is not a square array and should not be accepted by the system.

*The expected output was correct, the program returns the following string “ERROR!: Input must be square” passing this test case*

**Test Case 6:**

* + Input:
    - 0 0
    - 0 0
  + Expected Output: true
  + Reason: The sortable method should be able to handle scenarios where there is no variation in numbers

*The expected output was incorrect. Although the program was able to accept the values, it failed to identify if the table was sorted*

**Test Case 7:**

* + Input:
    - 1 2
    - 3 4
  + Expected Output: true
  + Reason: The sortable method should be able to handle scenarios where the table is already sorted

*The expected output was correct, the program was able to identify the sorted table, returning true passing this test case.*

**Test Case 8:**

* + Input:
    - 4 3
    - 2 1
  + Expected Output: true
  + Reason: The sortable method should be able to sort tables that are unsorted.

*The expected output was correct, the program was able to identify an unsorted table, then successfully sorting say table returning true, passing this test case*

**Test Case 9:**

NOTE: for this method, you must create a table with the size: INTEGER.MAX\_VALUE, or 2147483647. This will require you to create a table through code rather than importing it with a file. The input below is to be inserted prior to the sorting call in the main method. You may need to import a library as well.

* + Input:

Int[] maxArray = new int[INTEGER.MAX\_VALUE];

For(int i=0;i<INTEGER.MAX\_VALUE;i++){

maxArray[i] = I;

}

Table maxTable = new Table(INTEGER.MAX\_VALUE, maxArray);

* + Expected Output: True
  + Reason: This test case will test to ensure that the biggest possible NxN integer array can be tested.

*Unable to test this test case. Black box testing is used without the notion of the source code. The user should not be able to modify the code in any way, but to input values and expect a result*

**Test Case 10:**

NOTE: for this method, you must create a table with the size: INTEGER.MAX\_VALUE, or 2147483647. This will require you to create a table through code rather than importing it with a file. The input below is to be inserted prior to the sorting call in the main method. You may need to import a library as well.

* + Input:
    - 1
  + Expected Output: True
  + Reason: This test case will test to ensure that the smallest possible NxN integer array can be tested.

*Unable to test this test case. Black box testing is used without the notion of the source code. The user should not be able to modify the code in any way, but to input values and expect a result. Besides that, Test case 10 is repeating in someway to Test case 3 because it is testing the smallest value possible.*

**Test Case 11:**

* + Input:
    - 1 2
    - 3 4
  + Expected Output: true
  + Reason: The sortable method should be able to handle scenarios where the table is already sorted from a traditional left-to-right, top-to-bottom format.

*This test case is repetitive. Test case 7 was able to identify the output returning true and passing say test.*

**Test Case 12:**

* + Input:
    - 1 2
    - 1 2
  + Expected Output: true
  + Reason: The sortable method should be able to handle scenarios where the table is already sorted from a row-wise and column-wise format.

*The program can sort the input, but fails to identify if the table is sorted, it returns false, because the program sorted the table row-wise and column wise which is correct but fails this test case.*

**Test Case 13:**

* + Input:
    - 2 1
    - 3 4
  + Expected Output: false
  + Reason: The sortable method should return false in the scenario above since, even though it is sorted column-wise, it is not sorted row-wise on the top column. From this test case we can infer that the method will always check the top of the array.

*The way this program is code is to sort the table row-wise and column-wise. I believe the expected output for test case 13 should be true because it is not specified in the instructions how the program should sort the table. Although later instructions were given, the program does indeed sort the table returning true as specify in the original instructions.*

**Test Case 14:**

* + Input:
    - 0 1 2
    - 3 4 5
    - 8 7 5
  + Expected Output: false

Reason: The sortable method should return false in the scenario above since, even though it is sorted column-wise, it is not sorted row-wise on the bottom column. From this test case we can infer that the method will check the last row of the array meaning it must have checked each of the other rows as well

*The program returns true, because is failing to check if row is sorted. Failing this test case.*

**Test Case 15:**

* + Input:
    - 2 5
    - 1 7
  + Expected Output: false
  + Reason: The sortable method should return false in the scenario above since, even though it is sorted row-wise, it is not sorted col-wise on the bottom column. From this test case we can infer that the method will check the first column of the array.

*The program successfully returned false passing this test case*

**Test Case 16:**

* + Input:
    - 0 0 4
    - 1 1 3
    - 2 2 2
  + Expected Output: false
  + Reason: The sortable method should return false in the scenario above since, even though it is sorted row-wise, it is not sorted col-wise on the right column. From this test case we can see that the method will check the last column of the array meaning it must have checked each of the other columns as well.

*The program returns true because is checking that table is sorted row-wise, but not column-wise failing this test case.*

**The Test Plan Review:**

The test plan is describing a prior procedure before starting with the test. This is unnecessary, because the program that was tested has nothing to do with the code provided with the test plan. Test cases should be universal and focus on the functionality of the program. *The test cases were good, some were repetitive, but they were easy to follow and easy to understand. The instructions given, were clear and useful to achieve the desire outcome.*

**The Program Review:**

*The program was well managed. I was not able to test a file, but the program did sort the table and return true or false according if the table was sorted or not, then it proceeded to sort say table. The program checked if the table was sorted by column-wise, conflicting with some test cases, but it was able to sort the table column-wise, and row-wise. Passing most of the test cases.*