Table Sorter Test Plan

Notes prior to beginning:

Prior to beginning this test, please note that you will have to create separate test files (or modify the same test file) to contain the matrices found in the input. These matrices should be only numbers, unless otherwise specified, starting at the first character in the line with a space separating each number. The format will look as follows:

1 2

3 4

To utilize each of these files, you will have to manually change the java code that imports the file in the main method, as well as storing the test file in the same directory so that the java file is able to properly find it. This code is currently labelled as follows

**File hard = new File(“filename.txt”);**

You are of course welcome to create your own method that will run through all of the files or manually edit the file names themselves.

**Testing ability to handle malformed arrays**

This portion of testing is dedicated to ensuring that the program only allows square matrices to be sorted.

**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

**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.

**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.

**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

**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.

**Testing “sortable”**

This portion ensures that the method “sortable” is able to sort any arrays that it is tasked with sorting. Since the sortable method does not return anything, utilize the “isSorted” method to verify that the table was sorted.

**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

**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

**Test Case 8:**

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

**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.

**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.

**Testing “isSorted”**

This portion ensures that isSorted is returning the correct output based on the Table that it is asked to verify.

**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.

**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.

**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.

**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.

**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.

**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.