TableSort Test Plan

Test Objectives

This document outlines how to test the `isSorted` and `sortable` methods within the TableSorter class according to the following objectives:

- To test if the isSorted method accurately determines if a table is sorted in ascending order or not according to the below definition.
- To test if the sortable method accurately and reliably sorts a table into ascending order.

Test Criteria

A table is considered sorted if each row and each column is independently in ascending order. The following two tables are considered sorted:

```
Table 1
-----------
-1   2    7
8    11   15
18   22   31

Table 2
------------
-1   8    18
2    11   22
7    15   31
```

The isSorted method must meet the following criteria:

- Take in a Table object as an argument and return a boolean.
- Return True if the input table is sorted according to the above criteria.
- Return False if the input table is not sorted according to the above criteria.
- Return True for all empty tables.

The sortable method must meet the following criteria:

- Take in a Table object as an argument without returning anything.
- Rearrange the values in the table so that they meet the sorted criteria above.

Testing is complete when all of the below tests pass.

Testing Approach

To thoroughly test the TableSorter class, a variety of input tables need to be used. Each method should be tested independently using their corresponding input tables and the results analyzed according to the above criteria. To carry out each test, the given input table should be copied to a txt file and placed in the TableSorter directory. The following code can be used to test each method:

```
// Test isSorted method
try{
    Table t = Table.GetTable("test.txt");
    System.out.println(TableSorter.isSorted(t));
}catch(Exception e){
    System.out.println(e.toString());
}

// Test sortable method
try{
    Table t = Table.GetTable("test.txt");
    TableSorter.sortable(t);
    System.out.println(t.toString());
}catch(Exception e){
    System.out.println(e.toString());
}
```

The output of each test should then be analyzed according to the above criteria.

isSorted Inputs

The following inputs should be used to test the isSorted method. The output should be compared to the given output.

Basic Tests

Input	Expected Output	Testing for
12 14 19 34 89 100 104 109 200	True	Basic positive functionality
-89 -54 -23 -22 -5 109 19 100 200	True	Basic positive functionality
12 34 130 14 89 109 19 100 200	False	Basic negative functionality
74 71 23 49 10 4 83 12 119	False	Basic negative functionality

Edge Case Tests

Input	Expected Output	Testing for
12 14 19 34 89 100 104 109		

2147483647	True	Can handle largest posible int value at end
-2147483648 34 104 14 89 109 19 100 200	True	Can handle smallest possible int value at beginning
0 0 0 0 0 0 0 0 0	True	Can handle all equal values
12 34 130 -2147483648 89 109 19 100 200	False	Can handle smallest int value in middle
74 71 23 49 10 2147483647 83 12 119	False	Can handle largest int value in middle
212 34 130 14 89 109 19 100 200	False	Can handle largest value at beginning

Special Case Tests

Input	Expected Output	Testing for
8 8 8 11 42 76 97 98 99	True	Can handle row of equal values
17 23 97 17 27 101 17 36 108	True	Can handle column of equal values
0 0 0 0 0 0 0 0 0	True	Can handle all equal values
74 71 23 49 10 4 83 12 1	False	Can handle decending table

sortable Inputs

The following inputs should be used to test the sortable method. The output should then be validated using the definition of a sorted table given in Test Criteria section. There are multiple possible versions of a sorted table, any one of which is correct.

Basic Tests

Input	Testing for
12 34 130 14 89 109 19 100 200	Basic functionality
74 71 23 49 10 4 83 12 119	Basic functionality
-87 -32 15 88 21 232 -22 19 3	Basic functionality with negative numbers
12 14 19 34 89 100 104 109 200	Already sorted table should remain sorted
-89 -54 -23 -22 -5 109 19 100 200	Already sorted table should remain sorted

Edge Case Tests

Input	Testing for
12 34 130 -2147483648 89 109 19 100 200	Can handle smallest possible int
74 71 23 49 10 2147483647 83 12 119	Can handle largest possible int
12 34 130 14 89 109 19 100 20	Can handle last value out of place
212 34 130 14 89 109 19 100 200	Can handle first value out of place

Special Case Test

Input	Testing for
0 0 0 0 0 0 0 0 0	Can handle all the same value
-13 7 11 14 14 14 32 13 21	Can handle row of equal values
-13 31 11 14 31 41 32 31 21	Can handle column of equal values
74 71 23 49 10 4 83 12 19	Can handle table in reverse order