**Test Suite by <<author name redacted>>**  CS 5387

The purpose of this test suite is to conduct unit testing on the isSorted() method contained in the TableSorter Java class. This method verifies that every row and every column in a table is sorted in ascending order. The method sortTable() sorts every row and every column of a Table in ascending order. Conduct the following tests by creating a JUnit Java class, use the template below to construct this class.

ex. The Junit Java class below is names TableSorterTest. This class contains two methods, isSorted() and sortTable() to test the methods contained in the TableSorter Java class.

import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
  
class TableSorterTest {

@Test  
 void isSorted() throws Exception{

}  
  
 @Test  
 void sortTable() {  
  
 }  
}

Additionally, to conduct the tests it is necessary to create a TableSorter object.

All tests are conducted and written inside the isSorted() method. Array objects are created outside all test methods.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Identifier | Test Description | Test Data | Test Steps |
| Test #1 | Creates a 4 by 4 Table object with small size unsorted array with all positive 2-digit integers. Analyze the Table object before and after sorting. | {37, 28, 3, 38, 35, 2, 43, 5, 15, 32, 4, 19, 42, 30, 14, 16} | 1. Create array object with the test data. 2. Create Table object using the previously created array object inside the isSorted() test method. 3. Use assertEquals() method to verify the Table object is not sorted. 1 4. Perform a sorting operation on the table. 2 5. Use assertEquals() method to verify the Table object is now sorted. 3 |
| Test #2 | Create a 9 by 9 Table object with a medium size unsorted array with all positive 2- and 3-digit integers. Analyze the Table object before and after sorting. | {66, 47, 135, 181, 127, 74, 12, 9, 123, 140, 85, 8, 81, 187, 194, 129, 128, 71, 4, 84, 42, 39, 98, 68, 122, 91, 179, 44, 17, 56, 119, 101, 37, 59, 72, 130, 30, 108, 107, 133, 164, 174, 113, 54, 167, 16, 111, 131, 88, 197, 143, 86, 199, 11, 45, 60, 185, 48, 157, 15, 152, 114, 178, 189, 109, 13, 80, 103, 18, 158, 10, 36, 82, 172, 163, 136, 22, 126, 62, 49, 186, 96, 43, 146, 183, 102, 117, 75, 121, 125, 2, 160, 55, 33, 1, 193, 93, 198, 25, 46} | 1. Create array object with the test data 2. Create Table object using the previously created array object inside the isSorted() test method 3. Use assertEquals() method to verify the Table object is not sorted. 1 4. Perform a sorting operation on the table. 2 5. Use assertEquals() method to verify the Table object is now sorted. 3 |
| Test #3 | Creates a 10 by 10 Table object with a medium size sorted ( ascending order )array with all positive 2- and 3-digit integers. Analyze the Table object before and after sorting. | {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100} | 1. Create array object with the test data 2. Create Table object using the previously created array object inside the isSorted() test method 3. Use assertEquals() method to verify the Table object is sorted. 3 4. Perform a sorting operation on the table. 2 5. Use assertEquals() method to verify the Table object is still sorted. 3 |
| Test #4 | Creates a 10 by 10 Table object with medium size unsorted array with positive and negative multi-digit integers. Table will include smallest and highest values for an integer. Analyze the Table object before and after sorting. | {-2147483648, 0,-2000, 1, 4, 100, -3500, -10, 2147483647,2,4796440, 11632180, 5820329, 54189592, 57357722, 84805860, 90393156, 3448986, 81424166, 45157212, 63275932, 40455282, 65778777, 81664267, 53990038, 37970222, 98296117, 20041941, 93127710, 63754625, 6315223, 71451371, 2744234, 98762828, 42571177, 39816968, 60992391, 7625848, 19022843, 89504055, 69259314, 16846241, 56054059, 75987733, 26583307, 9725093, 71641291, 41814970, 30728236, 28550574, 21813774, 15722844, 49074898, 6860037, 93308727, 90265620, 87710564, 15028954, 17198072, 54003577, 50910564, 16007130, 79274908, 89347279, 43448120, 71394139, 11678649, 25835111, 71771656, 19261383, 42140903, 99893132, 70638572, 62790449, 46507905, 1005644, 41751052, 39347252, 91923868, 31224823, 59920751, 33960044, 54022385, 83903085, 52828391, 26229926, 76406246, 10595727, 5068337, 39730508, 5853641, 84304636, 61919765, 62482852, 86691345, 57255473, 28193504, 28398901, 53028542, 64566500, 80367126} | 1. Create array object with the test data 2. Create Table object using the previously created array object inside the isSorted() test method 3. Use assertEquals() method to verify the Table object is not sorted. 1 4. Perform a sorting operation on the table.2 5. Use assertEquals() method to verify the Table object is now sorted.3 |
| Test #5 | Creates a 5 by 5 Table object with small size same digit repetitions array with a positive 1-digit integers. Analyze the Table object before and after sorting. | {3,3,3,3,3,3,3,3,3,3  3,3,3,3,3,3,3,3,3,3,3  ,3,3,3,3} | 1. Create array object with the test data 2. Create Table object using the previously created array object inside the isSorted() test method 3. Use assertEquals() method to verify the Table object is sorted. 3 4. Perform a sorting operation on the table. 2 5. Use assertEquals() method to verify the Table object is still sorted.3 |
| Test #6 | Creates a 100 by 100 Table object with a file containing unsorted positive and negative multi-digit integers. Analyze the Table object before and after sorting. File must be located in same directory as testing class. | File Name:  TableFileTest.txt | 1. Create Table object using the TableFileTest.txt 2. Use assertEquals() method to verify the Table object is sorted. 1 3. Perform a sorting operation on the table. 2 4. Use assertEquals() method to verify the Table object is still sorted. 3 |

Executing Tests:

Compile and run the [TableSorterTestFileName].java file, after writing all tests, through command line or development software.

**Additional information:**

1 assertEquals(false, sorter.isSorted(tableX));

2 sorter.sortTable(tableX);

3 assertEquals(true,sorter.isSorted(tableX));

Ex. Using array x to create tableX with a square size of X by X.

Table tableX = new Table([X \* X],x);

Ex. Create Table object tableY with file.

Table tableY = Table.*GetTable*("FileName");

Ex. Conduct test #0 on tableX (sorter is an Object of class TableSorter)

/\*  
Test with table 0 before & after sorting  
 \*/  
 *assertEquals*(false, sorter.isSorted(tableX));  
 sorter.*sortTable*(tableX);  
 *assertEquals*(true,sorter.isSorted(tableX));

To verify Tables are sorted, a method to print the contents of a Table is available inside TableSorter class:

sorter.*printTable*(tableX);