<<author name redacted>>

Dr. Steven Roach

CS 4387: Software Verification and Validation

Black Box Test Plan – TableSort

Prerequisites

Before executing the following test plan, the tester should have the following two files on their machine:

1. Table.java (provided by Dr. Steven Roach via CS 4387 Piazza webpage)
2. TableSort.java (provided by Valente Arellano)

In order to proceed with testing, the tester should have an IDE/text editor (e.g. Eclipse, Atom, NetBeans, Notepad) and a Java SE Development Kit installed on their machine as they will be asked to write, compile and execute code in Java.

Test Overview

We will be testing two methods, both found within the TableSort class. Those two methods are **sortable**, which sorts a given Table object, and **isSorted** which verifies whether or not a given Table is sorted in ascending order.

The following test cases will be used to determine if the class methods are functioning correctly.

For isSorted(Table t):

* Given any 1x1 Table, isSorted is expected to return **true**.
* Given a sorted 2x2 Table, isSorted is expected to return **true**.
* Given an unsorted 2x2 Table, isSorted is expected to return **false**.
* Given any null Table object, isSorted is expected to return **false**.
* Given any Object of any datatype that is not Table, isSorted is expected to result in error.

For sortable(Table t):

* Given any 1x1 Table, sortable is expected to execute without altering the given Table (already sorted).
* Given any NxN Table with all cells containing the same integer value (e.g., {1, 1, 1, 1}), sortable is expected to execute without altering the given Table (already sorted).
* Given a 2x2 Table containing values {4, 3, 2, 1}, sortable is expected to sort the given Table like so: {1, 2, 3, 4}.
* Given a null Table object, sortable is expected to result in error.
* Given any Object of any datatype that is not Table, sortable is expected to result in error.

If all test cases above are passed, we can confidently say that the two methods within the TableSort class will function correctly for any NxN Table provided.

Instructions

1. Create a Java package to contain all classes used in this test. This package should be named “cs5387”.
2. Ensure that both Table.java and TableSort.java files are placed and saved in the package named “cs5387”.
3. Create a new Java class named “Test” or any other name you find appropriate. Ensure this class is also placed within the package named “cs5387”.
4. Create three empty plain text files, “values1.txt”, “values2.txt”, “values3.txt”, and ensure they are located in the package named “cs5387”.
   1. Within values1.txt, place any one value on one line (e.g., “1”);
   2. Within values2.txt, place the values {1, 1, 1, 1} on four separate lines.
   3. Within values3.txt, place the values {4, 3, 2, 1} on four separate lines.
5. In the Java class named “Test”, create a main method. All objects created in the following steps will be contained in this main method.
6. Declare one TableSorter object.
7. Create 3 Table objects (“t1”, “t2”, “t3”) by using the GetTable(filename) method. Provide the appropriate directories as the “filename” parameter (where your three plain text files are located, respectfully).
8. Create a fourth Table object (“t4”) that is null.
9. Execute the isSorted method on all 4 tables and print the results to the console. If all is functioning correctly:
   1. t1 should return true;
   2. t2 should return true;
   3. t3 should return false;
   4. t4 should return false;
10. Execute the sortable method on Table t1 ({1}). Print the table contents to the console after method is executed. If all is functioning correctly, the table contents should be unchanged.
11. Execute the sortable method on Table t2 ({1, 1, 1, 1}). Print the table contents to the console after method is executed. If all is functioning correctly, the table contents should be unchanged.
12. Execute the sortable method on Table t3 ({4,3,2,1}). Print the table contents to the console after method is executed. If all is functioning correctly, t3 should be sorted like so: {1, 2, 3, 4}.
13. Execute the sortable method on Table t4 ({null}). This method call should result in an error.
14. Lastly, create at least one more unsorted NxN table containing any values.
    1. Execute isSorted on this table and ensure the method returns **false**.
    2. Execute sortable on this table.
    3. Execute isSorted once more on this table and ensure the method returns **true**.

If all instructions were followed and no unexpected behavior occurred, the TableSorter class has officially passed this test.