BlackBox Homework Assignment

# Test TableSorter

The following is a test plan for the TableSorter Class. The TableSorter Class must implement a method called isSorted with the signature “public boolean isSorted(Table t)” that returns true if every row and every column of the table *t* is sorted in ascending order. TableSorter also requires a method called sortable with the signature “public static void sortable(Table t)” that sorts a table so that isSorted() is True.

The Blackbox tests will consist of tests that verify the correctness of the isSorted() method followed by tests that ensure that Tables sent to the sortable() method are returned sorted in such a way that isSorted() returns true.

The tests will be run using JUnit in eclipse under the assumption that the Table can only be created by using the GetTable(String filename) method from the Table class. This method stops all tables with non-integer values (alphabetical, null), greater than or less than the max integer values, or non-square tables from being created.

First Create a Junit test class by selecting new file -> Junit -> Junit .

A screenshot of a cell phone

Description automatically generated

Figure 1: Junit Test Case creation

## Test 1

**Objective:** Verify that isSorted returns true on a table where if the array was flattened the next number is always greater than the previous number.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 1 | | | | Current Status: Pending | | |
| Test title: Test sorted flattened array | | | | | | |
| Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable1.txt (Appendix: Figure 3) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable1.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestFlat()  Appendix: Figure 4. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedFlatTest method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 2

**Objective:** Verify that isSorted returns true on a table where the last number of a row is larger than the first number of the next column and all columns and rows are arranged in order from smallest to largest.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 2 | | | | Current Status: Pending | | |
| Test title: Test all rows and columns sorted in ascending order | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable2.txt (Appendix: Figure 7) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable2.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestNonFlat()  Appendix: Figure 8. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestNonFlat method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 3

**Objective:** Verify that isSorted returns true on a table where all numbers are the same.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 3 | | | | Current Status: Pending | | |
| Test title: Test all numbers are equal as sorted | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable3.txt (Appendix: Figure 9) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable3.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestAllEqual()  Appendix: Figure 10. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestAllEqual method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 4

**Objective:** Verify that isSorted returns true on a table that is length 1x1.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 4 | | | | Current Status: Pending | | |
| Test title: Test sorted if length 1x1 | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable4.txt (Appendix: Figure 11) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable4.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestAllEqual()  Appendix: Figure 12. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestAllEqual method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 5

**Objective:** Verify that isSorted returns false on a table where only the first number is out of order.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 5 | | | | Current Status: Pending | | |
| Test title: Test edge case of only the first number out of order | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable5.txt (Appendix: Figure 13) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable6.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestFirstWrong()  Appendix: Figure 14. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestFirstWrong method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 6

**Objective:** Verify that isSorted returns true on a table where only the last number is out of order.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 6 | | | | Current Status: Pending | | |
| Test title: Test only the last number out order | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable6.txt (Appendix: Figure 15) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable6.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestLastWrong()  Appendix: Figure 16. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestLastWrong method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 7

**Objective:** Verify that isSorted returns true on a table where the last number of a row is larger than the first number of the next column and all columns and rows are arranged in order from smallest to largest after the Table has gone through the sortable() method.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 7 | | | | Current Status: Pending | | |
| Test title: Test all rows and columns sorted in ascending order after sorting | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable2.txt (Appendix: Figure 7) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable2.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestNonFlatAfterSort()  Appendix: Figure 17. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestNonFlatAfterSort method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 8

**Objective:** Verify that isSorted returns true on a table where all numbers are the same after the Table has gone through the sortable() method.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 8 | | | | Current Status: Pending | | |
| Test title: Test all numbers are equal as sorted after sorting | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is required. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable3.txt (Appendix: Figure 9) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable3.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestAllEqualAfterSort ()  Appendix: Figure 18. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestAllEqualAfterSort method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 9

**Objective:** Verify that isSorted returns true on a table that is length 1x1 the Table has gone through the sortable() method.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 9 | | | | Current Status: Pending | | |
| Test title: Test sorted if length 1x1 after sorting | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is requires. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable4.txt (Appendix: Figure 11) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable4.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestAllEqualAfterSort ()  Appendix: Figure 19. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestAllEqualAfterSort method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 10

**Objective:** Verify that isSorted returns false on a table where only the first number is out of order but true after the Table has gone through the sortable() method.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 10 | | | | Current Status: Pending | | |
| Test title: Test edge case of only the first number out of order after sorting | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is requires. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable5.txt (Appendix: Figure 13) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable6.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestFirstWrongAfterSort ()  Appendix: Figure 20. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestFirstWrongAfterSort method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

## Test 11

**Objective:** Verify that isSorted returns true on a table where only the last number is out of order but true after the Table has gone through the sortable() method.

**Notes:** The project needs to be created in eclipse in order to follow these instructions exactly.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test No.: 11 | | | | Current Status: Pending | | |
| Test title: Test only the last number out order after sorting | | | | | | |
| Testing approach: Testing approach: The Eclipse IDE and understanding of how to create a Junit test case is requires. | | | | | | |
| STEP  Create the test file | OPERATOR ACTION  The test file testTable6.txt (Appendix: Figure 15) needs to be added to the to the project as shown in Appendix: Figure 2 | PURPOSE  The Junit test will require a file to read from | | | EXEPCTED RESULTS  The testTable6.txt is added to the project as shown in Appendix: Figure 1. | COMMENTS |
| Create the Junit test method | Create the test method isSortedTestLastWrongAfterSort ()  Appendix: Figure 21. | The Junit test case requires a method to test the interfaces in the TableSorter class. | | | The isSortedTestLastWrongAfterSort method is added to the Junit test case. |  |
| Run JUnit test class and verify all assertions are correct | Run the Junit test class as shown in Appendix: Figure 5.  Verify the Output is correct similar to the example shown in Appendix: Figure 6. | The Junit results will notify the tester of the success of the test and verify that isSorted works correctly for this test case. | | | As shown in Appendix: Figure 5. The results should have no Failures, and no Errors. |  |
| Concluding Remarks: | | | | | | |
| Testing Team:  N/A | | | Date Completed: | | | |

# Appendix

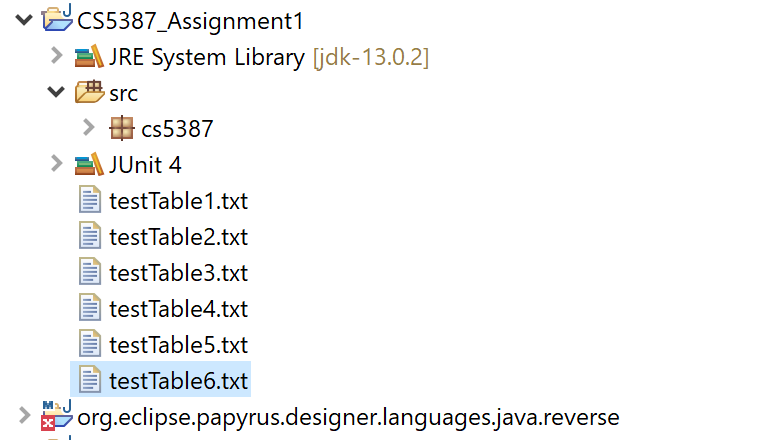


Figure 2: Test Table File examples

testTable1.txt

-50000 -25000 -2500 -450 0

10 11 800 900 9000

70000 78000 79000 80000 90000

990000 990005 990010 990048 990099

1990000 2990000 3990000 4990000 5990000

Figure 3: testTable1 File example

isSortedTestFlat ()

/\*\*

\* This method tests and verifies that a known sorted table is considered sorted by isSorted

\*\*/

@Test

**public** **void** isSortedTestFlat() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable1.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 4: isSortedTestFlat method example

A screenshot of a social media post

Description automatically generated

Figure 5: Run Selected Project

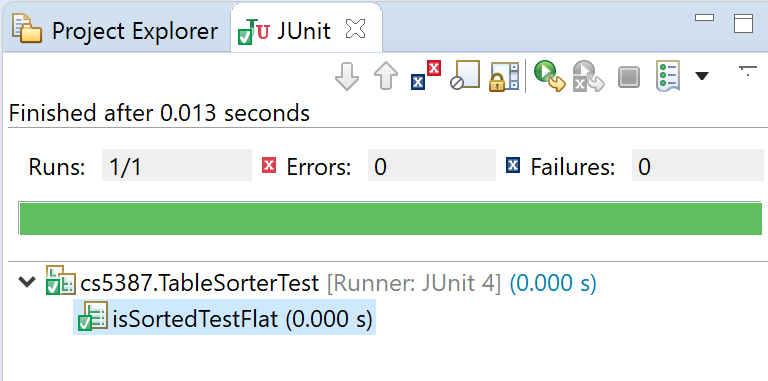


Figure 6: Example of JUnit Success

testTable2.txt

1 3 41

2 9 50

5 23 80

Figure 7: testTable2 File example

isSortedTestNonFlat()

/\*\*

\* This method tests and verifies that a known sorted table is considered sorted by isSorted

\*\*/

@Test

**public** **void** isSortedTestNonFlat() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable2.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 8: isSortedTestNonFlat method example

testTable3.txt

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

Figure 9: testTable3 File example

isSortedTestAllEqual()

/\*\*

\* This method tests and verifies that a table with all numbers equal is considered sorted by isSorted

\*\*/

@Test

**public** **void** isSortedTestAllEqual() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable3.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 10: isSortedTestAllEqual method example

testTable4.txt

5

Figure 11: testTable4 File example

isSortedTestSingle()

/\*\*

\* This method tests and verifies that a table of size 1x1 is considered sorted by isSorted

\*\*/

@Test

**public** **void** isSortedTestSingle() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable4.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 12: isSortedTestSingle method example

testTable5.txt

-1 0 1

-4 3 4

-2 5 8

Figure 131: testTable5 File example

isSortedFirstWrong ()

/\*\*

\* This method tests and verifies that a table with the first number out of order is recognized as unsorted

\*\*/

@Test

**public** **void** isSortedTestFirstWrong() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable5.txt");

*assertEquals*(**false**, sorter.isSorted(T1));

}

Figure 14: isSortedFirstWrong method example

testTable6.txt

-100 0 1

-20 3 4

-4 5 4

Figure 15: testTable6 File example

isSortedLastWrong ()

/\*\*

\* This method tests and verifies that a table with the last number out of order is recognized as unsorted

\*\*/

@Test

**public** **void** isSortedTestLastWrong() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable6.txt");

*assertEquals*(**false**, sorter.isSorted(T1));

}

Figure 16: isSortedLastWrong method example

isSortedTestNonFlatAfterSort()

/\*\*

\* This method tests and verifies that a known sorted table is considered sorted by isSorted after being sorted by sortable

\*\*/

@Test

**public** **void** isSortedTestNonFlatAfterSort() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable2.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

TableSorter.*sortable*(T1);

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 17: isSortedTestNonFlatAfterSort method example

isSortedTestAllEqualAfterSort ()

/\*\*

\* This method tests and verifies that a table with all numbers equal is considered sorted by isSorted after being sorted by sortable

\*\*/

@Test

**public** **void** isSortedTestAllEqualAfterSort() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable3.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

TableSorter.*sortable*(T1);

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 18: isSortedTestAllEqualAfterSort method example

isSortedTestSingleAfterSort ()

/\*\*

\* This method tests and verifies that a table of size 1x1 is considered sorted by isSorted after being sorted by sortable

\*\*/

@Test

**public** **void** isSortedTestSingleAfterSort() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable4.txt");

*assertEquals*(**true**, sorter.isSorted(T1));

TableSorter.*sortable*(T1);

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 19: isSortedTestSingleAfterSort method example

isSortedTestFirstWrongAfterSort ()

/\*\*

\* This method tests and verifies that a table with the first number out of order is recognized as sorted after being sorted by sortable

\*\*/

@Test

**public** **void** isSortedTestFirstWrongAfterSort() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable5.txt");

*assertEquals*(**false**, sorter.isSorted(T1));

TableSorter.*sortable*(T1);

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 20: isSortedTestFirstWrongAfterSort method example

isSortedTestLastWrongAfterSort ()

/\*\*

\* This method tests and verifies that a table with the last number out of order is recognized as sorted after being sorted by sortable

\*\*/

@Test

**public** **void** isSortedTestLastWrongAfterSort() **throws** FileNotFoundException, IOException, Exception {

TableSorter sorter = **new** TableSorter();

Table T1 = Table.*GetTable*("testTable6.txt");

*assertEquals*(**false**, sorter.isSorted(T1));

TableSorter.*sortable*(T1);

*assertEquals*(**true**, sorter.isSorted(T1));

}

Figure 20: isSortedTestLastWrongAfterSort method example