**5061 Test Results**

The following results are gathered using the tests in test plan 9914 and implementing them in code 5061.

**Test 1**

Checking the table can be generated for a square number of inputs

Input: 9 8

6 7

Result: The table with a square number of inputs is created

Function: In this test, the print statement and the output in the console proves that the table will be created if the number of inputs is square.

* System.***out***.println(t1.toString());
* [[I@d706f19]

9 8

7 6

**Test 2**

Checking the table will not be generated for a non-square number of inputs

Input: 9 8 7

6 5 4

Result: The table was not created, and an Exception error was issued by Junit and the console.

Function: In this test, the Exception errors prove that the table will not be created if the number of inputs is not square.

* java.lang.Exception (by Junit)
* ERROR!: Input must be square. (By Console)

**Test 3**

Checking the table can be generated with positive integers

Input: 5 7

9 8

Result: The table was created with positive integers

Function: In this test, the print statement and the output in the console proves that the table will be created if the input has positive integers.

* System.***out***.println(t3.toString());
* [[I@7ee955a8]

5 7

9 8

**Test 4**

Checking the table can be generated with negative integers

Input: -9 -8

-4 -3

Result: The table was created with negative integers

Function: In this test, the print statement and the output in the console proves that the table will be created if the input has negative integers.

* System.***out***.println(t4.toString());
* [[I@7ee955a8]

-9 -8

-4 -3

**Test 5**

Checking the table can be generated with two positive integers, zero, and a negative integer

Input: 5 1

0 -3

Result: The table was created with two positive integers, a zero, and a negative integer

Function: In this test, the print statement and the output in the console proves that the table will be created if the input has positive and negative integers, and a zero.

* System.***out***.println(t5.toString());
* [[I@48fa0f47]

5 1

1. -3

**Test 6**

Checking the method isSorted() returns true for a sorted table

Input: -1 0

2 3

Result: The method isSorted() returned true

Function: In this test, assertTrue is used to check the method isSorted() returns true for the input table.

* *assertTrue*(TableSorter.*isSorted*(t6));

**Test 7**

Checking the method isSorted() returns false for an unsorted table

Input: 1 9 2

8 0 7

-5 4 6

Result: The method isSorted() returned false and failed the test.

Function: In this test, assertTrue is used to check the method isSorted() returns false, and fails the test. Thus proving that isSorted() returns false for unsorted tables.

*assertTrue*(TableSorter.*isSorted*(t7));

**Conclusion:**

The test cases 1-5 evaluate Table, and test 6-7 evaluate the method isSorted() from TableSorter. Test cases 3-5 could be used to test sortable().

Test cases that evaluated the method isSorted() in TableSorter passed, ensuring that the method was fully functional. However, the functionality of method sortable is yet to be tested.