Table Sorter Program

<<author name redacted>>

Test plan

Version <1.0>

02/10/20

Table of Contents

[1. Introduction 3](#_Toc32416847)

[1.1. Purpose 3](#_Toc32416848)

[2. Test Items and Other Items 4](#_Toc32416849)

[2.1. Method to be tested 4](#_Toc32416850)

[2.2. Types of Input 4](#_Toc32416851)

[2.3. Files 4](#_Toc32416852)

[3. Test Cases 5](#_Toc32416853)

[3.1. Test case 1: 5](#_Toc32416854)

[3.2. Test case 2: 5](#_Toc32416855)

[3.3. Test case 3: 5](#_Toc32416856)

[3.4. Test case 4: 5](#_Toc32416857)

[3.5. Test case 5: 5](#_Toc32416858)

[3.6. Test case 6: 5](#_Toc32416859)

[3.7. Test case 7: 6](#_Toc32416860)

[4. How To Test 7](#_Toc32416861)

[4.1. Testing the Program: 7](#_Toc32416862)

[4.2. Ways to Run Test(s): 7](#_Toc32416863)

[**4.2.1. Run All Tests 7**](#_Toc32416864)

[**4.2.2. Run Individual Tests 8**](#_Toc32416865)

# Introduction

## Purpose

The program to be tested has the functionality of sorting an N x N Table. The program sorts the every row and every column in ascending order. The following are the main methods of the program:

* isSorted(): method in which returns True if every row and every column of the Table is sorted in ascending order.
* sortable(): method in which sorts a Table so that isSorted() becomes true.

The testing approach of this Test Plan is to use JUnit as our primary way to test the functionality of the program with various test cases.

# Test Items and Other Items

## Method to be tested

* isSorted(): method in which returns True if every row and every column of the Table is sorted in ascending order.
* sortable(): method in which sorts a Table so that isSorted() becomes true.

## Types of Input

* *File***:** File containing a list of numbers separated by spaces.
* *Initialized Array***:** Array with initialized values.

## Files

* Table.java
* TableSorter.java
* TableSorterTest.java
* Input.txt
* Table3.txt
* Table5.txt
* Table6.txt

# Test Cases

## Test case 1:

* **Input**: file with all negative numbers
* **Objectives**: Test the functionality of the program with negative numbers.
* **Expected Output**: Sorted Table with sorted rows and columns

## Test case 2:

* **Input**: sorted array with ascending order values
* **Objectives**: Test the functionality of the program with an already sorted Table
* **Expected Output**: Sorted Table with sorted rows and columns

## Test case 3:

* **Input**: sorted array with same numbers
* **Objectives**: Test the functionality of the program with input containing same value
* **Expected Output**: Sorted Table with sorted rows and columns

## Test case 4:

* **Input**: initialized array with random numbers
* **Objectives**: Test the functionality of the program with input containing random numbers both negative and positive.
* **Expected Output**: Sorted Table with sorted rows and columns

## Test case 5:

* **Input**: initialized array with only ‘0’ and ‘1’
* **Objectives**: Test the functionality of the program with input containing only zeroes and ones.
* **Expected Output**: Sorted Table with sorted rows and columns

## Test case 6:

* **Input**: initialized array with random numbers
* **Objectives**: Test the functionality of the method isSorted() with an unsorted Table
* **Expected Output**: False

## Test case 7:

* **Input**: file containing 100 numbers (10x10)
* **Objectives**: Testing the functionality of the program with large input.
* **Expected Output**: Sorted Table with sorted rows and columns

# How to Test

## Testing the Program:

* In order to test this program, Junit5 was implement in the file ‘TableSorterTest.java’. This file contains all the test cases shown in Section 3.

## Ways to Run Test(s):

### Run All Tests

* Open up program files with Integrated Development Environment (e.g. InteliJ, Eclipse)
* Right click on the file ‘TableSorterTest.java’ and run this file, as shown in the following image. A close up of text on a black background

  Description automatically generated
* If successful, the following should appear on your *Run* and *Event Log* window.

A screenshot of a computer screen

Description automatically generated

* The capture above depicts which test cases passed and which did not. In the ‘Event Log’ it shows the number of tests that passed.

### Run Individual Tests

* Open up program files with Integrated Development Environment (e.g. InteliJ, Eclipse)
* Each test case is in an individual method in the file ‘TableSorterTest.java’.
* Travel to desired test case method and right click on the method and chose run, as shown in the following image.

A screenshot of a cell phone screen with text

Description automatically generated

* Once method is ran, the following should appear on your ‘Run’ window showing if method has passed.

A screenshot of a cell phone

Description automatically generated

* This can be repeated to any other method within the file ‘TableSorterTest.java’