Table of Contents

[Project Approach 1](#_Toc114348118)

[Below is the order in which I built this assignment: 1](#_Toc114348119)

[Testing & Refactoring 2](#_Toc114348120)

[Test Cases: 2](#_Toc114348121)

[loadFile Cases: 2](#_Toc114348122)

[Setup for Load File Case: 3](#_Toc114348123)

[TEST CASE 1: Load input.txt file – **TEST CASE PASSED** 3](#_Toc114348124)

[TEST CASE 2: Load empty.txt file – **TEST CASE FAILED** 3](#_Toc114348125)

[**RETEST**: TEST Case 2: Load empty.txt file – **TEST CASE PASSED** 3](#_Toc114348126)

[TEST CASE 3: Load header.txt – **TEST CASE PASSED** 3](#_Toc114348127)

[TEST CASE 4: Load 100.txt – **TEST CASE PASSED** 3](#_Toc114348128)

[addRow Test Cases: 2](#_Toc114348129)

[Setup for AddRow tests: 2](#_Toc114348130)

[**TEST CASE 1:** Add row after loading input.txt – **TEST CASE PASSED** 2](#_Toc114348131)

[**TEST CASE 2:** Add row after loading 100.txt – **TEST CASE FAILED** 2](#_Toc114348132)

[**RETEST TEST CASE 2**: Add row after loading 100.txt – TEST CASE PASSED 2](#_Toc114348133)

[Remove Row Test Cases 3](#_Toc114348134)

[**TEST CASE 1:** Remove a row from 100.txt – **TEST CASE FAILED** 3](#_Toc114348135)

[**TEST CASE 1:** Remove a row from 100.txt – **TEST CASE FAILED** 3](#_Toc114348136)

[**RETEST TEST CASE 1:** Remove a row from 100.txt – **TEST CASE PASSED** 3](#_Toc114348137)

[End to End Test 3](#_Toc114348138)

[**TEST CASE 1:** Add row after removing a row from 100.txt. Then find the added row. – **TEST CASE PASSED** 3](#_Toc114348139)

# Project Approach

I feel like I tend to crawl through my projects/assignments bit by bit and testing as I go. I've left some of my testing outputs commented out in the TextTable class that will show some of the small things I'm testing along the way. Overall, I find myself not being very confident with my skills yet and still feeling quite rusty after the summer break.

## Below is the order in which I built this assignment:

1. Load the .jar
2. Create TextTable and TableRow class
3. Create the skeleton of each class based on assignment instructions
4. Build out loadTableFromFile
5. Build out saveTabletoFile
6. Build out addRow - *this one gave me a lot of trouble in my understanding of the utilization of the TableRow class*
7. Build out findRow - *this one gave me a lot of trouble in nailing down the exact type of loop I should be using to get through the array. I'm also still not sure that my approach with converting the array row to a string and using string splitter was even the right thing to do.*
8. Build out Remove Row

# Testing & Refactoring

I decided midway through find row that I was using split string a lot and created an additional method in TextTable to deal with that.

## Test Cases:

### loadFile Cases:

#### Setup for Load File Case:

1. Create multiple files and place in AssignementOne directory:
   1. **input.txt** - the file provided in the assignment
   2. **empty.txt** - a blank file
   3. **header.txt** - a file with just a header
   4. **100.txt** - a file with 100 rows after the header
2. Create Validation println:
   1. The following println in loadTableFromFile will validate the header and the row count as it progresses through the file:  
      System.***out***.println("TEST HEADER: " + headerInfo + " TEST CURRENT ROW #: " + numRows);
   2. The following println in loadTableFromFile will validate the fullTable array after each line is added:

System.***out***.println(Arrays.*deepToString*(fullTable));

* 1. The following println in addRow (called from loadTable) will validate the individual row as it’s being added:

System.***out***.println(singleRow.toString());

#### TEST CASE 1: Load input.txt file – **TEST CASE PASSED**

1. **Expected results:** no errors, and the print ln statements will confirm that the table is being created.
2. **Actual Results** (sample from console): **As Expected**Calendar

   Description automatically generated with medium confidence
3. After confirming file load, Exit program

#### TEST CASE 2: Load empty.txt file – **TEST CASE FAILED**

1. Expected results: no errors
2. Actual results: **TEST CASE FAILED**. Unhandled No Such Element Exception was thrown.

#### **RETEST**: TEST Case 2: Load empty.txt file – **TEST CASE PASSED**

Setup: Ahead of testing, add a new try/catch ahead of the headerInfo=inFile.next();.

1. Expected results: If it is unable to find any line at all, it will get caught and the user will get an exception message in the console indicating that the file selected was empty.
2. Actual results: **As Expected.**

Text

Description automatically generated

1. After confirming file load, Exit Program

#### TEST CASE 3: Load header.txt – **TEST CASE PASSED**

1. Expected results: no errors, println statements will not show any array/table data.
2. Actual results: **As expected.**
3. After confirming file load, exit program.

#### TEST CASE 4: Load 100.txt – **TEST CASE PASSED**

1. Expected results: no errors, println statements will show the full array/table with all 100 elements. The final row in the file is ‘100 Jen100 199.1’
2. Actual results: **As expected** (screenshot below of last println)

A picture containing text

Description automatically generated

1. After confirming file load, exit program.

### addRow Test Cases:

#### Setup for AddRow tests:

Ensure the println statements in the addRow method are not commented out.

#### **TEST CASE 1:** Add row after loading input.txt – **TEST CASE PASSED**

1. Add the following: 66; Jen; 18.5
2. Expected Results: Println will confirm that a row 8 has been added and display the correctly values.
3. Actual Results: As expected.

Text

Description automatically generated

1. After confirming, quit program

#### **TEST CASE 2:** Add row after loading 100.txt – **TEST CASE FAILED**

1. Add the following: 66; Jen; 18.5
2. Expected Results: Likely an exception, but unsure where to handle it. Expect this case to fail.
3. Actual Results: **FAILED. ArrayIndexOutOfBoundsException**

#### **RETEST TEST CASE 2**: Add row after loading 100.txt – TEST CASE PASSED

1. Add the following: 66; Jen; 18.5
2. Expected Results: Exception will be handled, and user will be advised the table is full in the console.
3. Actual Results: As expected



### Remove Row Test Cases

#### **TEST CASE 1:** Remove a row from 100.txt – **TEST CASE FAILED**

1. Remove row 50 by entering the ID ‘50’ in the inputDialog.
2. Expected results: row 50 will be removed and the rows in the table will be moved, with row 100 being left as null
3. Actual Results: **FAILED**. IndexOutofBoundsException on line 79.

#### **TEST CASE 1:** Remove a row from 100.txt – **TEST CASE FAILED**

1. Remove row 50 by entering the ID ‘50’ in the inputDialog.
2. Expected results: row 50 will be removed and the rows in the table will be moved, with row 100 being left as null
3. Actual Results: **FAILED**. IndexOutofBoundsException on line 79.

#### **RETEST TEST CASE 1:** Remove a row from 100.txt – **TEST CASE PASSED**

After much staring at the code and getting a second set of eyes, I had a typo in my for loop. I forgot the second &.

1. Remove row 50 by entering the ID ‘50’ in the inputDialog.
2. Expected results: row 50 will be removed and the rows in the table will be moved, with row 100 being left as null
3. Actual Results: **PASSED**

A picture containing text, file

Description automatically generated

### End to End Test Cases

#### **TEST CASE 1:** Add row after removing a row from 100.txt. Then find the added row. – **TEST CASE PASSED**

1. Load 100.txt
2. Remove row 100
3. Add a row: 200; Mike;18.5
4. Find row with name ‘Mike’
5. Expected results: Will find the row Mike
6. Actual Results: **PASSED**.

TEST CASE 2: Load File (header.txt), Add rows, remove rows, Find Rows, Save Table.

1. Load header.txt.
2. Add 4 distinct rows:
   1. 40 Jen 9.20
   2. 8 Ira 5.5
   3. 25 Celia 2.12
   4. 0 Luka 9.14
3. Remove row 25
4. Add row 25 Celia 2.12
5. Find row ‘Luka’
6. Save File as Family.txt & review file system
7. Actual Results:
   1. Row found:

Graphical user interface, application, Word, website

Description automatically generated

* 1. File Created:

A picture containing text

Description automatically generated

1. Actual Results: **PASSED**