Contents

[How I implemented/tested… 2](#_Toc119234822)

[CityTable Class 2](#_Toc119234823)

[StadiumTable Class 2](#_Toc119234824)

[Overall Testing 2](#_Toc119234825)

[Using a Map class 3](#_Toc119234826)

[Test Cases 3](#_Toc119234827)

[TestCase1: City Table Walkthrough 3](#_Toc119234828)

[Description: 3](#_Toc119234829)

[Steps: 3](#_Toc119234830)

[Expected Outcomes: 3](#_Toc119234831)

[Actual Outcome: 3](#_Toc119234832)

[TestCase2: Stadium Table Walkthrough 3](#_Toc119234833)

[Description: 3](#_Toc119234834)

[Steps: 4](#_Toc119234835)

[Expected Outcomes: 4](#_Toc119234836)

[Actual Outcomes: 4](#_Toc119234837)

[TestCase3: City/Stadium Tables in the same run 4](#_Toc119234838)

[Description: 4](#_Toc119234839)

[Steps: 4](#_Toc119234840)

[Expected Outcome: 4](#_Toc119234841)

[Actual Outcome: 5](#_Toc119234842)

[TestCase4: Load Empty Files 5](#_Toc119234843)

[Description: 5](#_Toc119234844)

[Steps: 5](#_Toc119234845)

[Expected Outcomes: 5](#_Toc119234846)

[Actual Outcomes: 5](#_Toc119234847)

[TestCase5: Add and locate cities composed of multiple strings 5](#_Toc119234848)

[Description: 5](#_Toc119234849)

[Steps: 5](#_Toc119234850)

[Expected outcomes: 6](#_Toc119234851)

[Actual outcomes: 6](#_Toc119234852)

[TestCase6: Load a city table of 100 rows 6](#_Toc119234853)

[Description: 6](#_Toc119234854)

[Steps: 6](#_Toc119234855)

[Expected Outcome: 6](#_Toc119234856)

[Actual Outcome: 6](#_Toc119234857)

[TestCase7: Load a stadium table of 100 rows 6](#_Toc119234858)

[Description: 6](#_Toc119234859)

[Steps: 6](#_Toc119234860)

[Expected Outcome: 7](#_Toc119234861)

[Actual Outcome: 7](#_Toc119234862)

# How I implemented/tested…

## CityTable Class

In order to iteratively implement and test the changes I made to use List/ArrayList, I started by solely working with the CityTable class. I commented out the bulk of the methods in StadiumTable to not get errors at run time. Additionally, I new I needed to only work w/ one method at a time in the city table, so the first method I \*had\* to fix first was addRow. This was a simple enough. This allowed me to run the program and make sure load & add worked without throwing errors.

Though I knew there were no errors, and I could tell through debugging things were working, I wanted the visual proof by saving down/exporting the results to a new text. From here I moved to saveTable. I updated the println to use fullTable.get(i) to make sure I was printing one row at a time. Tested the save table and made sure my output files were reflecting adds.

Then I finished up by tackling findRow and removeRow. Testing this piece included adding the row, finding the row I added, then removing it. Finally ending by saving the table.

I also did some boundary testing – now that I’m using List/ArrayList, I can blow up the files past 100 row limit. I decided to **not** implement a 100 row restriction manually to take full advantage of the capabilities.

## StadiumTable Class

With CityTable done, I moved on to StadiumTable and repeated what I had done in CityTable.

## Overall Testing

The test cases below were pulled from Assignment3 testing. I followed the same test strategy to ensure the program worked as it did previously.

## Using a Map class

Okay so full disclosure, I really felt like I was understanding collection and interfaces and generics in the lecture and in the reading. But connecting the dots between lecture topic and using it w/ our abstract class was really tough. I felt like I was “guessing” and doing trial/error rather than implanting with full confidence. I’m not actually quite sure if I implemented things as you expected? That said, this assignment I feel like I could sort of understand why using HashMap would be better (the CityId could be the “primary” key potentially in a key value pair). But I was really confused because it’s a key value PAIR – so in CityTable, there are 2 other elements besides cityID and StadiumTable there are 3 other elements. So not a pair? So I think I’m not truly understanding I *could*  refactor this to using HashMap.

# Test Cases

## TestCase1: City Table Walkthrough

### Description:

This test case will validate that a City Table can be loaded, rows can be added, the new row can be found, old rows can be found, a row can be deleted, and the table can be saved to an output file.

### Steps:

1. Load cities.txt - Option 1
2. Add a row of city data (Sparta, 700, 1.2) – Option 3
3. Locate the new row – Option 5
4. Locate a pre-existing row – Option 5
5. Delete the first row – Option 4
6. Save the table as citiesTest1.txt – Option 2
7. Quit (Option 11)

### Expected Outcomes:

1. File will load without errors
2. Row will be added without errors
3. Will locate the new row
4. Will locate the existing row
5. Row will get deleted and the remaining rows will get re-indexed
6. File will save to the project folder with the new row at the end of the list and the first row removed.
7. Program will close.

### Actual Outcome:

1-7: As expected

## TestCase2: Stadium Table Walkthrough

### Description:

This test case will validate that a Stadium Table can be loaded, rows can be added, the new row can be found, old rows can be found, a row can be deleted, and the table can be saved to an output file.

### Steps:

1. Load stadia.txt - Option 6
2. Add a row of city data (Jen's Stadium, 72, Jen’s Team, 7000) – Option 8
3. Locate the new row – Option 10
4. Locate a pre-existing row – Option 10
5. Delete the first row (Mercedes-Benz Stadium)– Option 9
6. Save the table as stadiaTest2.txt – Option 7
7. Quit (Option 11)

### Expected Outcomes:

1. File will load without errors
2. Row will be added without errors
3. Will locate the new row
4. Will locate the existing row
5. Row will get deleted and the remaining rows will get re-indexed
6. File will save to the project folder with the new row at the end of the list and the first row removed.
7. Program will close.

### Actual Outcomes:

1-7: As expected

## TestCase3: City/Stadium Tables in the same run

### Description:

This test case will hit all the commands in the program in the same run.

### Steps:

1. Load cities.txt - Option 1
2. Load stadia.txt - Option 6
3. Add a row of city data (Holmen, 50, .05) – Option 3
4. Add a row of stadium data (Another Stadium, 50, Another Team, 50) – Option 8
5. Locate the new city row – Option 5
6. Locate the new stadium row – Option 10
7. Locate a pre-existing city row – Option 5
8. Locate a pre-existing stadium row – Option 10
9. Delete the first city row – Option 4
10. Save the table as citiesTest3.txt – Option 2
11. Delete the Soldier Field (ID 18)– Option 9
12. Save the table as stadiaTest3.txt – Option 7
13. Quit (option 11)

### Expected Outcome:

1. City File will load without errors
2. Stadium File will load without
3. City Row will be added without errors
4. Stadium Row will be added without errors
5. Will locate the new city row
6. Will locate the new stadium row
7. Will locate the existing city row
8. Will locate the existing stadium row
9. City Row will get deleted and the remaining rows will get re-indexed
10. File will save to the project folder with the new row at the end of the list and the first row removed.
11. Stadium row will be removed and the remaining rows will get re-indexed
12. File will save to the project folder with the new row at the end of the list and the first row removed.
13. Program will close.

### Actual Outcome:

1-13: As expected

## TestCase4: Load Empty Files

### Description:

This test case will validate that users will get a warning message if they try to load an empty city table or empty stadium table.

### Steps:

1. Load empty.txt as a city table – Option 1
2. Load empty.txt as a stadium table – Option 6
3. Quit (Option 11)

### Expected Outcomes:

1. User will get an error message
2. User will get an error message
3. Program will close

### Actual Outcomes:

1-3: As expected

## TestCase5: Add and locate cities composed of multiple strings

### Description:

This test will validate that city names with multiple worders (ex. La Crosse, West Salem, etc.) are added to the table correctly, can be found, and saved to a file without losing formatting.

### Steps:

1. Add a row of city data (La Crosse, 700, .05) – Option 3
2. Add a row of city data (West Salem, 800, .005) – Option 3
3. Add a row of city data (New York City, 1, 1.2) – Option 3
4. Locate La Crosse – Option 5
5. Locate 800 – Option 5
6. Locate 1– Option 5
7. Save the table as citiesTest5.txt – Option 2

### Expected outcomes:

1. City will get added without error
2. City will get added without error
3. City will get added without error
4. City will be found
5. City will be found
6. City will be found
7. Table will get saved and the 3 cities will be in the file in the correct format.

### Actual outcomes:

1-7: As expected

## TestCase6: Load a city table of 100 rows

### Description:

This test case will validate that we can load a full city table of 100 rows and manipulate it and save it. A test file call 100City.txt has been staged to work with

### Steps:

1. Load 100City.txt – Option 1
2. Add a row of city data (Jen, Jen, Jen) – Option 3
3. Remove a row of city data (Seattle92, 92, 91.77) – Option 4
4. Add a row of city data (Ira, Ira, Ira) – Option 3
5. Add another row of city data (James, James, James) – Option 3
6. Save the table as citiesTest6.txt – Option 2
7. Quit (Option 11)

### Expected Outcome:

1. File will load without issues/errors
2. **Without the 100 row max, row will get added to the array.**
3. The row will get removed and the remaining rows re-indexed
4. The new row will get added
5. The new row will get added
6. The final file will get saved to the project folder with the deleted row missing and the row added. **It will have 102 rows.**

### Actual Outcome:

1-6: As expected

## TestCase7: Load a stadium table of 100 rows

### Description:

This test case will validate that we can load a full stadium table of 100 rows and manipulate it and save it. A test file call 100Stadia.txt has been staged to work with

### Steps:

1. Load 100Stadia.txt – Option 6
2. Add a row of stadium data (Jen, Jen, Jen, Jen) – Option 8
3. Remove a row of stadium data (109/BBVA Stadium98)– Option 9
4. Add a row of stadium data (Ira, Ira, Ira, Ira) – Option 8
5. Save the table stadiaTest7.txt – Option 7
6. Quit (Option 11)

### Expected Outcome:

1. File will load without issues/errors
2. **Without the 100 row max, row will get added to the array.**
3. The row will get removed and the remaining rows re-indexed
4. The new row will get added
5. The final file will get saved to the project folder with the deleted row missing and the rows added. **It will have 101 rows.**

### Actual Outcome:

1-6: As expected