Assignment 2: Hash Tables

(replacement for P3c & P4a)

DUE: Apr. 19th at 11:59pm

Extra Credit Available for Early Submissions!

Setup

- Download the `assignment2.zip` and unzip it.
- This will create a folder `section-yourGMUUserName-a2`.
- Rename the folder replacing `section` with the `001`, `002`, `005`, etc. based on the lecture section you are in.
- Rename the folder replacing `yourGMUUserName` with the first part of your GMU email address.
- After renaming, your folder should be named something like: `001-krusselc-a2`.
- Complete the `readme.txt` file (an example file is included: `exampleReadmeFile.txt`).

Submission Instructions

- Make a backup copy of your user folder!
- Remove all test files, jar files, class files, etc.
- You should just submit your java files and your readme.txt
- Zip your user folder (not just the files) and name the zip `section-username-a2.zip` (no other type of archive) following the same rules for `section` and `username` as described above.
  - The submitted file should look something like this:

```
001-krusselc-a2.zip --> 001-krusselc-a2 --> JavaFile1.java
                                             JavaFile2.java
                                             ...
```

- Submit to blackboard. DOWNLOAD AND VERIFY WHAT YOU HAVE UPLOADED THE RIGHT THING. ***Submitting the wrong files will result in a 0 on the assignment!***

Basic Procedures

You must:

- Have code that compiles with the command: `javac *.java` in your user directory without errors or warnings.
- Have code that runs with the command: `java ThreeTenHashTable1`, and `java ThreeTenHashTable2`, and `java DemoProgram` in your user directory.
- Have a style (indentation, good variable names, etc.) -- you must pass the style checker!
- Comment your code well in JavaDoc style -- you must pass the comments checker!

You may:

- Add additional methods and variables to both provided classes, however these methods **must be private**.

You may NOT:

- Make your program part of a package.
- Add additional *public* methods or variables.
- Add any additional libraries/packages which require downloading or use any code from the internet.
- Import any additional libraries/packages or add any additional import statements (or use the "fully qualified name" to get around adding import statements). THERE SHOULD BE NO IMPORTS IN ANY FILES.
- Alter any method/class signatures defined in this document of the template code. Note: "throws" is part of the method signature in Java, don't add/remove these.
- Alter provided classes or methods that are complete (e.g. **`DemoProgram`**, **`toString()`**, etc.).
- Add `@SuppressWarnings` to any methods unless they are private helper methods for use with a method we provided which already has an `@SuppressWarnings` on it.

Grading Rubric

No Credit

- Non submitted assignments
- Assignments submitted after 5pm the Monday after the due date
- Non-compiling assignments
- Non-independent work
- Code that violates and restrictions or "you may not" mandates.
- "Hard coded" solutions
- Code that would win an obfuscated code competition with the rest of CS310 students

How will my assignment be graded?

- Automatic Testing (100%): To assess the correctness of programs.
- You CANNOT get points for code that doesn't compile or for submitting just the files given to you.
- Extra credit for early submissions:
  - 1% extra credit rewarded for every 24 hours your submission made before the due time
  - Up to 5% extra credit will be rewarded
  - Your latest submission before the due time will be used for grading and extra credit checking. You CANNOT choose which one counts.

Automated Testing Rubric

The JUnit tests used for grading will NOT be provided for you (you need to test your own programs!), but the tests will be based on what has been specified in the project description and the comments in the code templates. A breakdown of the point allocations is given below:

50 pts	ThreeTenHashTable1 – Open Addressing with Linear Probing
50 pts	ThreeTenHashTable2 – Separate Chaining
-5pts ("off the top")	**Not following the submission format** Note: This is very, *very* important for these assignments; the graders need to return grades *very* fast. If you do not follow the submission format (the same one you've been using for P0, P1, and P2), then they will manually deduct 5pts from your score. No exceptions.
-5 pts ("off the top")	Not passing code style check
-5 pts ("off the top")	Not passing JavaDoc style check

"Off the top" points (i.e. items that will lose you points rather than earn you points).

Assignment Overview

You will be creating two different types of hash maps (hash tables that "map" unique keys to values). One map (**`ThreeTenHashTable1`**) will use open addressing with linear probing and the other (**`ThreeTenHashTable2`**) will use separate chaining. Both maps will *do the same things* (i.e. they will have the same operations, such as **`put()`**, **`get()`**, **`rehash()`**, etc.), but they will *do them in different ways* (probing vs chaining).

The storage (**`storage`**) for each has been setup for you and you cannot change this. For open addressing, the storage is **`TableEntry<K,V>[]`** while for separate chaining the storage is **`Node<K,V>[]`**. The **`TableEntry`** class is defined in **`TableEntry.java`** (it's a simple key-value pair class) and the **`Node`** class (which uses the **`TableEntry`** class) is defined in **`ThreeTenHashTable2`**.

DO NOT TRY TO "BLIND CODE" THIS PROJECT! You need to understand exactly the storage you have and how to use it, if you do this, the project should be very straight forward, but otherwise this will be a very, *very* difficult.

Tasks Breakdown and Sample Schedule

There are **2** tasks in this assignment. It is suggested that you implement these tasks in the given order:

- Task 1: Write a hash table using open addressing with linear probing (50%)
- Task 2: Write a hash table using separate chaining (50%)

Need a schedule?

- You've got 1.5 weeks.
- There are 15 methods to write.
- You have other classes with exams/projects.
- Assume you want to spend the last half week getting EC or seeking additional help.
- Keeping those things in mind, fill in the following:
    - 3/23-3/26: ____ (first week period)
        - Suggestions: Finish zyBooks and textbook readings about hash tables, start *designing*.
    - 3/27-3/29: ____ (first weekend period)
        - Suggestions: Tasks 1 and 2 (implement using your *design* from earlier)
    - 3/30-4/02: ____ (second week period)
        - Suggestions: *Thorough* Testing and Debugging
    - 4/03-4/05: ____ (second weekend period)
        - Suggestions: Turn in early for Extra Credit

Examples for Testing

There are 11 "yays" in the main methods of both class tables, but we have also created a DemoProgram which allows you to interactively add elements to a map

The remainder of this document contains two sample runs of the demo program with annotations. ***Make sure you get the exact same output when you do the project.*** If your maps look different, then something's wrong in your implementation.

Note: In the example runs on the next few pages, **`user input`** is shown in **`green highlight and underlined`**.

Example Runs of DemoProgram for Table Type 1

```
>java DemoProgram 1

This is a demo interactive program
Be aware that both the keys and va
Strings, so if you enter 1 as your
string "1" not the integer 1.
```

← Runs the main method for the demo program, requesting to use the first type of table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 10
Added value at key.
(Hit <Enter> to Continue)
```

← Adds a key-value pair to the table.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: banana:10
******************************************
(Hit <Enter> to Continue)
```

← Shows the current table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 1
Updated value at key.
(Hit <Enter> to Continue)
```

← Updates the value associated with the key if the key is in the table.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: banana:1
******************************************
(Hit <Enter> to Continue)
```

← Shows the current table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 2
----------Getting a Value by Key----------
Enter a key: banana
Associated value is 1
(Hit <Enter> to Continue)
```

← Request the value associated with the given key.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: banana:1
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 3
----------Removing a Key-Value Pair----------
Enter a key: banana
Removed pair was (banana,1)
(Hit <Enter> to Continue)
```

← Removes the banana, note it says what was removed.

```
******************************************
Table Size: 0, Capacity: 2
[0]: null
[1]: tombstone
******************************************
(Hit <Enter> to Continue)
```

← Table has a tombstone!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 2
----------Getting a Value by Key----------
Enter a key: banana
No such key
(Hit <Enter> to Continue)
```

← Confirmed, no banana in the table.

```
******************************************
Table Size: 0, Capacity: 2
[0]: null
[1]: tombstone
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 1
Added value at key.
(Hit <Enter> to Continue)
```

← Put the banana-1 pair back in.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: banana:1
******************************************
(Hit <Enter> to Continue)
```

← Tombstone replaced!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: orange
Enter a value: 2
Added value at key.
(Hit <Enter> to Continue)
```

← Add some more fruit…

```
******************************************
Table Size: 2, Capacity: 4
[0]: null
[1]: null
[2]: orange:2
[3]: banana:1
******************************************
(Hit <Enter> to Continue)
```

← Load too high when adding, table rehashed.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: pear
Enter a value: 3
Added value at key.
```

← Add some more fruit…

```
(Hit <Enter> to Continue)

******************************************
Table Size: 3, Capacity: 4
[0]: pear:3
[1]: null
[2]: orange:2
[3]: banana:1
******************************************
(Hit <Enter> to Continue)
```

← Load doesn't require expanding the table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: peach
Enter a value: 3
Added value at key.
(Hit <Enter> to Continue)
```

← Added more fruit...

```
******************************************
Table Size: 4, Capacity: 8
[0]: null
[1]: peach:3
[2]: orange:2
[3]: banana:1
[4]: null
[5]: null
[6]: pear:3
[7]: null
******************************************
(Hit <Enter> to Continue)
```

← Bigger table...

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 10
Resized table
(Hit <Enter> to Continue)
```

← Manually resize table to a larger table.

```
******************************************
Table Size: 4, Capacity: 10
[0]: orange:2
[1]: null
[2]: null
[3]: peach:3
[4]: pear:3
[5]: null
[6]: null
[7]: banana:1
[8]: null
[9]: null
******************************************
(Hit <Enter> to Continue)
```

← Items moved around!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 5
Resized table
(Hit <Enter> to Continue)
```

← Manually resize table to a smaller table...

```
******************************************
Table Size: 4, Capacity: 5
[0]: orange:2
[1]: null
[2]: banana:1
[3]: peach:3
[4]: pear:3
******************************************
(Hit <Enter> to Continue)
```

← Works as long as there is at least one open spot in the table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 4
Unable to resize table to requested size
(Hit <Enter> to Continue)
```

← Try to resize without enough room… does not alter the table.

```
******************************************
Table Size: 4, Capacity: 5
[0]: orange:2
[1]: null
[2]: banana:1
[3]: peach:3
[4]: pear:3
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 1
Unable to resize table to requested size
(Hit <Enter> to Continue)
```

← Same problem.

```
******************************************
Table Size: 4, Capacity: 5
[0]: orange:2
[1]: null
[2]: banana:1
[3]: peach:3
[4]: pear:3
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: carrot
Enter a value: 3
Added value at key.
(Hit <Enter> to Continue)
```

← Add another food.

```
******************************************
Table Size: 5, Capacity: 10
[0]: orange:2
[1]: null
[2]: null
[3]: peach:3
[4]: pear:3
[5]: carrot:3
[6]: null
[7]: banana:1
[8]: null
[9]: null
******************************************
(Hit <Enter> to Continue)
```

← Load too high, table expands.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 6
```

← Exit program.

__Example Runs of DemoProgram for Table Type 2__

```
>java DemoProgram 2

This is a demo interactive prog
Be aware that both the keys and
Strings, so if you enter 1 as your key, you get the
string "1" not the integer 1.
```

← Runs the main method for the demo program, requesting to use the ***__second__*** type of table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 10
Added value at key.
(Hit <Enter> to Continue)
```

← Adds a key-value pair to the table.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: [banana:10]->null
*******************************
(Hit <Enter> to Continue)
```

← Shows the current table, this one is an **__array of linked lists__**!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 1
Updated value at key.
(Hit <Enter> to Continue)
```

← Updates the value associated with the key if the key is in the table.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: [banana:1]->null
******************************************
(Hit <Enter> to Continue)
```

← Updated table.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 2
----------Getting a Value by Key----------
Enter a key: banana
Associated value is 1
(Hit <Enter> to Continue)
```

← Request the value.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: [banana:1]->null
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 3
----------Removing a Key-Value Pair----------
Enter a key: banana
Removed pair was (banana,1)
(Hit <Enter> to Continue)
```

← Removes the banana.

```
******************************************
Table Size: 0, Capacity: 2
[0]: null
[1]: null
******************************************
(Hit <Enter> to Continue)
```

← **No tombstones in separate chaining!**

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 2
----------Getting a Value by Key----------
Enter a key: banana
No such key
(Hit <Enter> to Continue)
```

← Confirmed, no banana.

```
******************************************
Table Size: 0, Capacity: 2
[0]: null
[1]: null
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: banana
Enter a value: 1
Added value at key.
(Hit <Enter> to Continue)
```

← Put the banana-1 pair back in.

```
******************************************
Table Size: 1, Capacity: 2
[0]: null
[1]: [banana:1]->null
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: orange
Enter a value: 2
Added value at key.
(Hit <Enter> to Continue)
```

← Add some more fruit…

```
******************************************
Table Size: 2, Capacity: 4
[0]: null
[1]: null
[2]: [orange:2]->null
[3]: [banana:1]->null
******************************************
(Hit <Enter> to Continue)
```

← Load too high, rehashed.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: pear
Enter a value: 3
Added value at key.
```

← Add some more fruit…

```
(Hit <Enter> to Continue)

******************************************
Table Size: 3, Capacity: 4
[0]: null
[1]: null
[2]: [orange:2]->null
[3]: [banana:1]->[pear:3]->null
******************************************
(Hit <Enter> to Continue)
```

← Load doesn't require expanding the table, but **some "chains" are now forming**!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: peach
Enter a value: 3
Added value at key.
(Hit <Enter> to Continue)
```

← Add more fruit...

```
******************************************
Table Size: 4, Capacity: 8
[0]: null
[1]: [peach:3]->null
[2]: [orange:2]->null
[3]: [banana:1]->null
[4]: null
[5]: null
[6]: null
[7]: [pear:3]->null
******************************************
(Hit <Enter> to Continue)
```

← Bigger table, shorter chains...

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 10
Resized table
(Hit <Enter> to Continue)
```

← Manually resize table to a larger table.

```
******************************************
Table Size: 4, Capacity: 10
[0]: [orange:2]->null
[1]: null
[2]: null
[3]: [peach:3]->[pear:3]->null
[4]: null
[5]: null
[6]: null
[7]: [banana:1]->null
[8]: null
[9]: null
******************************************
(Hit <Enter> to Continue)
```

← Items moved around, *and* **some chains occurred** even with the bigger table!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 5
Resized table
(Hit <Enter> to Continue)
```

← Manually resize table to a smaller table...

```
******************************************
Table Size: 4, Capacity: 5
[0]: [orange:2]->null
[1]: null
[2]: [banana:1]->null
[3]: [peach:3]->[pear:3]->null
[4]: null
******************************************
(Hit <Enter> to Continue)
```

← Works as long as there is **at least one slot**!

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
----------Resizing the Table----------
Enter a new size: 4
Resized table
(Hit <Enter> to Continue)
```

← Resizing smaller, **still works**!

```
******************************************
Table Size: 4, Capacity: 4
[0]: null
[1]: [peach:3]->null
[2]: [orange:2]->null
[3]: [banana:1]->[pear:3]->null
******************************************
(Hit <Enter> to Continue)


Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 4
```

```
----------Resizing the Table----------
Enter a new size: 1
Resized table
(Hit <Enter> to Continue)
```

← Even smaller, **still works**!

```
******************************************
Table Size: 4, Capacity: 1
[0]: [peach:3]->[orange:2]->[banana:1]->[pear:3]->null
******************************************
(Hit <Enter> to Continue)
```

↖ All in one chain. Note the order of items here! **Very important that it comes out the same for you!** If it doesn't, you're not following the instructions in the rehash method properly.

```
Options:
        1. Add/Replace a Ke
        2. Get the value a
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 1
----------Adding/Updating a Key-Value Pair----------
Enter a key: carrot
Enter a value: 3
Added value at key.
(Hit <Enter> to Continue)
```

← Add another food.

```
******************************************
Table Size: 5, Capacity: 8
[0]: null
[1]: [peach:3]->null
[2]: [orange:2]->null
[3]: [banana:1]->null
[4]: null
[5]: [carrot:3]->null
[6]: null
[7]: [pear:3]->null
******************************
(Hit <Enter> to Continue)
```

← Load too high *after* adding, table does a "double in size and rehash" ***until* the load is below 80%!** If the items are in the wrong order, you may be trying to expand all at once instead of rehashing repeatedly.

```
Options:
        1. Add/Replace a Key-Value Pair
        2. Get the value associated with a key
        3. Remove a key
        4. Resize the table
        5. Display the table
        6. Quit
Enter a menu choice: 6
```

← Exit program.