02. Hash Table Implementation

Work Individually

You must develop the code for this assignment entirely on your own. If you encounter difficulties while working, you are encouraged to discuss general algorithms and debugging strategies with anyone you would like. If you feel that getting assistance will require someone else viewing your code, the course staff are the only people that you are allowed to share or show any part of your code with prior to the late deadline for this assignment. You may not store or share your code in any way that another students can access. And you are not allowed to view or make use of any Hash Table code that is not written entirely by you.

HashTableMap Specifications

You must define a public class HashTableMap that implements **this provided MapADT interface (http://pages.cs.wisc.edu/~cs400/MapADT.java)** . Your implementation must:

- be left in the default package: ie. do not define it within a named package.
- be defined with two generic type parameters in this order: <KeyType, ValueType> which allow it to be used with different types of key and value pairings.
- include two constructors with the following signatures:
  - `public HashTableMap(int capacity)`
  - `public HashTableMap() // with default capacity = 10`
- use a private array instance field within your HashTableMap class to store key-value pairs.
- grow by doubling capacity and rehashing, whenever its load capacity becomes greater than or equal to 85%. Define private helper methods to organize this functionality.
- store new values in your hash table at the index corresponding to the (absolute value of your key's hashCode()) modulus the HashTableMap's current capacity. When the put method is passed a key that is null or is already in your hash table (use .equals() to check for this equality), that call should return false without making any changes to the hash table. The put method should only return true after successfully storing a new key-value pair.

- include a remove method that returns a reference to the value associated with the key that is being removed. When the key being removed does not exist, this method should instead return null.
- include a size method that returns the number of key-value pairs stored in this collection, and include a clear method that removes all key-value pairs from this collection.
- use chaining to handle hash collisions. You may make use of the java.util.LinkedLists class for this purpose.
- throw exceptions as indicated within the MapADT interface. You should make use of java.util.NoSuchElementException for this.
- **NOT** make use of any classes from java.util other than the two exceptions listed above: you may use java.util.LinkedList, and java.util.NoSuchElementException.

Assignment Submission

Please submit your progress on this assignment early and often to **gradescope (https://gradescope.com/)** as a form of backup, and to get feedback from the automated grading tests. If you do not have access to the gradescope assignment page notify a member of the course team or join using the following code: "WYD7K7". The two files that you should include with your submissions are: HashTableMap.java and MapADT.java (as provided, no edits necessary). If you define extra classes in other source files as part of your solution, you should include those in your upload too. Your most recent submission in gradescope will be marked "active" by default, but you can change this to an earlier on-time or less buggy submission if you prefer. Ensure that your final "active" submission for this assignment is clearly organized, consistently styled, well documented with comments, and includes the following file header information at the top of each and every source file:

```
// --== CS400 File Header Information ==--
// Name: <your full name>
// Email: <your @wisc.edu email address>
// Notes to Grader: <optional extra notes>
```

Grading Rubric

Criteria	Assignment	Points
Formative Gradescope Tests	Individual	15
Summative Gradescope Tests	Individual	15
Code Clarity (Manual)	Individual	5
Total Points Possible		**35**