Homework 5: HashMaps

Due: See T-Square

Important

There are general submission guidelines you must always follow. If you fail to follow any of the following guidelines you risk receiving a **0** for the entire assignment.

1. All submitted code must compile under **JDK 8**. This includes unused code, so don't submit extra files that don't compile.
2. Do not include any package declarations in your classes.
3. Do not change any existing class headers, constructors, or method signatures.
4. Do not add additional public methods when implementing an interface.
5. Do not use anything that would trivialize the assignment. (e.g. don't import/use `java.util.LinkedList` for a Linked List assignment. Ask if you are unsure.)
6. You must submit your source code, the `.java` files, not the compiled `.class` files.
7. All methods must be efficient, even if an expected runtime is not specified.
8. Do not add any new instance variables.
9. After you submit your files redownload them and run them to make sure they are what you intended to submit. You are responsible if you submit the wrong files.
10. The submission on T-Square **MUST** contain only files you want us to grade. **Remove any old versions of files, and submit only the latest files.**

Hash Map

In this homework, you will implement a key-value Hash Map with a quadratic probing collision policy. A Hash Map maps keys to values and allows $O(1)$ average case lookup of a value when the key is known. This Hash Map must be backed by an array of initial size 11, and must have a size of $2n + 1$ when the table exceeds (greater than, not greater than or equal to) a load factor of 0.67. These values are provided as constants in the interface and should be used within your code.

The table should not accept duplicate keys, but **duplicate values are acceptable.**

Neither keys nor values may be null.

Hash functions

You are not expected to write your own hash functions for this assignment; use the `hashCode()` method (every `Object` has one). If this is a negative value, use the absolute value of it.

Quadratic Probing

Your HashMap must implement a quadratic probing collision policy. If the hash value of the key is occupied, probe in increments based on a quadractic. For example, if the hash value of your key is 3 with a backing array of size 11, and index 3 in the array is occupied, check index $3 + 1 \mod 11$, then $3 + 4 \mod 11$, then $3 + 9 \mod 11$, etc. A good visualization can be found here. If you check n iterations, where n is the size of the backing array, and no open spot was found, regrow the table, and then add the data.

Exceptions

When throwing exceptions, you must include a message by passing in a String as a parameter. For example:

```
throw new PDFReadException("Did not read PDF, will lose points.");
```

Style and Formatting

It is important that your code is not only functional but is also written clearly and with good style. We will be checking your code against a style checker that we are providing. It is located in resources along with instructions on how to use it. We will take off a point for every style error that occurs. If you feel like what you wrote is in accordance with good style but still sets off the style checker please email Jonathan Jemson (jonathanjemson@gatech.edu) with the subject header of "CheckStyle XML".

Javadocs

Javadoc any helper methods you create in a style similar to the Javadocs for the methods in the interface.

Forbidden Statements

You may not use these in your code at any time in CS 1332.

- `break` may only be used in switch-case statements
- `return` may not be used in void methods
- `continue`
- `package`
- `System.arrayCopy()`
- `clone()`
- `assert()`
- `Arrays` class
- `Array` class
- `Collections` class
- Reflection APIs

If you use these, we will take off points.

Debug print statements are fine, but should not print anything when we run them. We expect clean runs - printing to the console when we're grading will result in a penalty.

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them.

1. `HashMap.java` This is the class in which you will actually implement the interface. Feel free to add private helpers but **do not add any new public methods.**

2. `HashMapInterface.java` This is the interface your HashMap should implement. All instructions for what the methods should do are in the Javadoc. **Note that what you return for the `toArray()` method should be the entire backing array, including the empty spaces. Do not alter this file.**

3. `MapEntry.java` A class representing an entry in a HashMap. **Do not alter this file.**

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder, copy over the interfaces, recompile, and run. It is your responsibility to re-test your submission and discover editing oddities, upload issues, etc. **Submit only the files you want us to grade, and remove any old versions of files. There should be exactly ONE file for each of the files listed below.**

1. `HashMap.java`

You may attach each file individually or submit them in a zip archive.