|  |  |
| --- | --- |
| **Project #6** | **CMSC 132** |
| **Due:  Friday 10/28 at 11:00 pm** | **Object-Oriented Programming II** |
| **Type of project: Closed** | **Fall 2016** |

**Dense Bag**

****

**Introduction**

"Bag" is an abstract data type that is like a set, but allows duplication of elements. By "Dense Bag" we mean a bag that is capable of *efficiently* handling cases where some of the elements are repeated a huge number of times. For example, if the bag contains 100,000,000 copies of a particular item then you would *not* want to store 100,000,000 references to the item. Think about how you could store information about the number of copies of each element that are in the bag without actually storing multiple copies.

**What You Will Do**

You will write a class called DenseBag, which will extend the AbstractCollection class. We will be employing Java generics, so your DenseBag can be used to store any type of objects the user desires. The DenseBag must support the "iterator" method, and so you will also be writing an inner class implementing an iterator over the DenseBag.

**Note: You must implement the remove method for the Iterator this time!**

### Data Structure?

You get to decide how to store the data. Make a good choice, or you will fail tests! When the user adds elements to the collection, you must record how many of each element have been added, without actually storing multiple copies.

### Implementation Details

In a Bag, removing an item removes a single instance of the item. For example, a Bag b could contain additional instances of the String "a" even after calling b.remove("a").

The iterator for a dense bag must iterate over all instances, including duplicates. In other words, if a bag contains 5 instances of the String "a", an iterator will generate the String "a" 5 times. You must implement the remove method for the iterator. It should remove a single instance of the last element that was returned by a call to "next".

In addition to the methods defined in the Collection interface, the DenseBag class supports several additional methods: uniqueElements, getCount, addMany, and choose.

The class extends AbstractCollection in order to get implementations of addAll, removeAll, retainAll and containsAll. (We will not be over-riding those). All other methods defined in the Collection interface will be implemented here.

For more specific implementation details, [please read the javadoc.](http://docs.google.com/doc/index.html)

**Grading**

The public tests, release tests, and secret tests together will comprise 90% of your grade on the project. The remaining 10% will be "style grading", done by visual inspection. Be sure to use good variable names, proper indentation and use of braces, etc. Also be careful to avoid redundant code.

[Web Accessibility](https://www.umd.edu/web-accessibility)