CSC 207 **Assignment 8: Multiset** **Spring 2011**

Assigned: Tuesday 12 April 2011

Due: Tuesday 19 April 2011

Objectives:

- Increase fluency in **inheritance** and **polymorphism** by developing a class hierarchy
- Continue practice with the Java **Collections** Framework
- Practice developing **inner classes**
- Reinforce the use of **type bounds** in **generics**
- Refine **unit testing** skills for composite objects and methods with non-primitive return values

Collaboration: This homework assignment may be completed individually or in pairs.

Submission: Follow the instructions for submitting programs via P-Web and handing in a printed copy. Be sure to generate ***ample*** unit tests and a statement justifying their sufficiency.

Implementation

In mathematics, a multiset is a set that allows repeated elements. Like a set, it has no intrinsic ordering of its elements. The file `~weinman/courses/CSC207/code/multiset/MultiSet.java` contains a specification for a multiset interface. A simple, but not very efficient implementation has been given in `LinkedMultiSet.java` (in the same location). This is not efficient because it stores repeated elements multiple times. You will provide more efficient implementations of this data structure.

`AbstractMultiSet`

Create an abstract class called `AbstractMultiSet` that *completely* implements the `MultiSet` interface. The only unimplemented method should be the constructor.

Hint: Declare a polymorphic member variable whose ultimate type depends on the concrete implementation (below) but provides the necessary methods.

`HashMultiSet`

Create a concrete `HashMultiSet` class that extends `AbstractMultiSet`. The only method implemented in this class should be the constructor. All the available methods that result should have constant time performance on average.

`TreeMultiSet`

Create a `TreeMultiSet` class that extends `AbstractMultiSet`. The only method implemented in this class should be the constructor. All the available methods that result should have logarithmic time performance on average. In addition, the iterator should provide the elements in their natural ordering.

Testing

To test the iterator of your `TreeMultiSet`, you can use the `Tester` class's `checkIterable` method. However, the `HashMultiSet`'s `iterator` does not give any guarantees on the ordering. Therefore, you'll have to be more creative in testing its correctness.

Hint: Think about how you can verify all the items in the multiset appeared the correct number of times during iteration.

Application

Using the concrete multiset of your choice, write a program that takes the name of a file on the command line and generates a histogram of word counts. That is, print each "word" (a token separated by whitespace) along with the number of times it appears.