Assignment 2 Word Frequency with Linked Lists

Due Date: Wednesday Feb 3, before midnight
Weight: 5%
This assignment should be completed individually.

Description

This assignment is a continuation of assignment 1. The behaviour of your program should be identical to assignment 1, the only difference is that you cannot use an `ArrayList` to store your `Word` objects, you must use a singly linked list of your own implementation.

The finished program will be run like this:

```
cat input.txt | java A1.jar > output.txt
```

`input.txt` is any text file containing English text.

The output from your program must be *Exactly* as shown in the examples.

Implementation Details

- Create a class called `A2`. It will have the `main()` method and do the bulk of the processing. Be sure to set it up to instantiate an `A2` object.
- Input is all from standard input. Create a `Scanner` object and use that for all input:

  ```
  Scanner inp = new Scanner(System.in);
  ```

  All output should be to standard output. Use `System.out.print();` and `System.out.println();` for all output.
- You have a `Word` class from assignment 1 to represent a word. It should implement `Comparable`, should implement equals and should include two `Comparator` objects that order from highest to lowest frequency and vice versa. You should not need to alter it.
- You should create a linked list class called `SLL`. You can use the `Node` class given out in lab as a starting point. If you create methods in `SLL` that match the ones in `ArrayList`, you can use much of your existing code with little editing.

Your `Node` class uses a generic type for the data that it stores. However, it should not just be any old class. You want to be able to order your list, so you need a type that implements `Comparable`. That is done like this:

```
public class Node<T extends Comparable<T>>
```

and

```
public class SLL<T extends Comparable<T>>
```

This will restrict `T` to be only classes that implement `Comparable`.

- The easiest way to do this assignment is to simply replace the `ArrayList<Word>` of words with a `SLL<<Word>` object.
- Initially you should only worry about building a list in alphabetical ordering.
- To find the top ten most frequent or least frequent words I want you to create two more lists. Build these lists from the alphabetically sorted list, but order it by word frequency, from highest to lowest, then create another list that orders from lowest to highest word frequency.

  You should only ever have one instance of each `Word` object, so a given `Word` object will be part of all three lists.
- Use these three lists to print out your results as in Assignment 1.
- Testing and example files. There are seven sample input and output pairs named `inp1.txt`, `out1.txt` etc. Use these to determine if your program is running correctly. You should use the `diff` command to compare your output to the samples.

  ```
  diff (cat sampleout.txt) (cat myout.txt)
  ```

  If `diff` gives no output, your program is working correctly.

  You should also devise a test plan and create a series of test files to fully exercise your program. For example, what about an empty file? or one with only some punctuation in it? Or a single word repeated 100 times? or?

  I will be evaluating the program against files you have not seen yet that will be meant to *test* your code.
- This is a fairly short assignment. You really only need four classes, `A2`. `Word`, `Node` and `SLL`. Focus on writing compact, efficient code.

Documentation and Coding Standards

Your program should follow all of the coding rules and guidelines outlined in the document provided. In particular, include `Javadoc` style comments for all classes and substantial methods.

What to Hand In

Hand in a single file, `A2.jar`.

Submit this to the Blackboard drop box provided. The jar should be executable and contain:

1. All of your `.class` and `.java` files.
2. A class called `A2` which has a `main()` method.

Grading

A detailed grading rubric will be provided.

I will run your program with the command line given above on three text files and compare your output to the specifications.

Outcomes

Once you have completed this assignment you will have:

- Implemented an ordered linked list;
- Used the Java Comparable interface;
- Used a Comparator object;
- Used standard input and standard output re-direction.