CS1316 HW6: Shakespeare Analysis

DUE DATE: NOON on Monday, July 7th, 2008 with a 3 hour grace period.

I know we are all fascinated by the works of William Shakespeare. The homework is to study two ways to solve the same problem – finding the frequency of occurrence of each word the playwright used in a given passage.

Eventually, we will provide a test program that will read in arbitrarily large amounts of Shakespeare text. A typical example is attached to this assignment as Mercy.txt. What we want is a printout of all the words used in the passage in alphabetical order without listing duplicated words more than once, but with the number of times each word occurs in the given text.

You will be required to solve this problem two ways: first using the built-in LinkedList class, and second by inserting each word as a String into a BST.

Infrastructure

You will need to write a Word class that contains a String named 'word' and an int named 'frequency' and extends Comparable. For convenience, the data can be public. It will need:

1. a Constructor that is given the word and sets the frequency to 1, and
2. a compareTo method that compares your Word to a String, returning -1, 0 or 1. Hint: the String class already has the compareTo you need.
3. A toString method that prints the word and its frequency.
4. A test main to verify this behavior.

LinkedList Approach

You will need to write the LLShakespeare class with the following capabilities:

1. A constructor that initializes an empty LinkedList<Word>.
2. A toString method that will traverse that list to print out each word and its frequency.
3. An insert method that consumes a word and inserts it into the Linked List in alphabetical order. If the word is already in the list, just increment its frequency.
4. An add method that consumes a line of Shakespeare and breaks it into words using the StringTokenizer class. [See the Java API for how to use this

class.] The delimiters for the tokenizer should include all known punctuation except the apostrophe (‘).

5. A main program that calls your add method repeatedly with sample strings taken from Mercy.txt like the following:

   “The quality of mercy is not strain'd,”

   “It droppeth as the gentle rain from heaven”

   “Upon the place beneath: it is twice blest;”

   and then invokes your toString method to list the words and their frequencies.

BST Approach

To your horror, you discover that when you put large amounts of text into your analysis, it runs much too slowly. You have to re-implement your analysis (keeping most of it intact) using a BST of Word objects instead of your LinkedList<Word>. Make a copy of your LLShakespeare code and save it as a BSTShakespeare class. If you wrote the LLShakespeare class well, you should have to change only the insert and toString methods. Even your test main program should need only a type change from LLShakespeare to BSTShakespeare.

Remember, if you find the string already in the tree, you should increment the word’s frequency.

Requirements:

- Make sure your homework compiles before turning it in.
- Make sure to turn in the .java support files necessary to run it.

How to Turn In

Turn in only the following files via TSquare:

- Word.java,
- LLShakespeare.java, and
- BSTShakespeare.java