interface WordFrequencyList:

```java
/**
 * An alphabetized list of words and associated frequencies of occurrence in
 * a text
 *
 * @author Russell C. Bjork
 * @version March 22, 2008
 */

public interface WordFrequencyList
{
    /** Record the occurrence of a word in the text
     *
     *  @param word the word that occurred in the text
     *  @return a modified list in which either the word is added at the
     *          appropriate place with frequency of 1 (if it had not occurred
     *          before), or the frequency of an existing word is increased by
     *          one.
     */
    WordFrequencyList recordOccurrence(String word);

    /** Report the total length of this list
     *
     *  @return the length of this list
     */
    int length();

    /** Report the total frequencies for all words in this list
     *
     *  @return the total frequencies for all words in this list
     */
    int totalFrequencies();

    /** Report the number of occurrences of a particular word
     *
     *  @param word the word
     *  @return the number of times this word has occurred
     */
    int occurrencesFor(String word);

    /** Print all words in this list with their associated frequencies
     */
    void print();

    /** Convert words to all uppercase
     *
     *  @return a fresh list with the same words but each in all uppercase
     *  letters
     */
    WordFrequencyList capitalize();
}
```

class EmptyWordFrequencyList:

```
/**
 * An empty word frequency list
 *
 * @author Russell C. Bjork
 * @version March 22, 2008
 */

public class EmptyWordFrequencyList implements WordFrequencyList
{
    /**
     * Constructor for objects of class EmptyWordFrequencyList
     */
    public EmptyWordFrequencyList() {
    }

    // The following methods are specified by interface WordFrequencyList

    public WordFrequencyList recordOccurrence(String word) {
         return new NonEmptyWordFrequencyList(word, 1, this);
    }

    public int length() {
        return 0;
    }

    public int totalFrequencies() {
        return 0;
    }

    public int occurrencesFor(String word) {
        return 0;
    }

    public void print() {
    }

    public WordFrequencyList capitalize() {
        return this;
    }
}
```

class NonEmptyWordFrequencyList:

```java
/**
 * A word frequency list containing one or more words
 *
 * @author Russell C. Bjork
 * @version March 22, 2008
 */

public class NonEmptyWordFrequencyList implements WordFrequencyList
{
    /**
     * Constructor for objects of class NonEmptyWordFrequencyList
     *
     * @param firstWord the firstWord in this list
     * @param firstWordFrequency the frequency for this word
     * @param rest the rest of this list
     */
    public NonEmptyWordFrequencyList(String firstWord,
                                     int firstWordFrequency,
                                     WordFrequencyList rest) {
        this.firstWord = firstWord;
        this.firstWordFrequencyCount = firstWordFrequency;
        this.rest = rest;
    }

    // The following methods are specified by interface WordFrequencyList

    public WordFrequencyList recordOccurrence(String word) {

        if (firstWord.equals(word)) {
            // The word matches the first word in this list - create
            // a new list with modified frequency for this word, and the
            // same rest
            return new NonEmptyWordFrequencyList(firstWord,
                                                 firstWordFrequencyCount + 1,
                                                 rest);
        }
        else if (firstWord.compareTo(word) < 0) {
            // The word is after the first word in this list - create
            // a new list containing the same first word, but with a rest
            // that records the modified count for the word
            return new NonEmptyWordFrequencyList(firstWord,
                                                 firstWordFrequencyCount,
                                                 rest.recordOccurrence(word));
        }
        else {
            // The word belongs before the first word in this list - hence
            // needs to be added before rest of list
            return new NonEmptyWordFrequencyList(word, 1, this);
        }
    }

    public int length() {
        int restLength = rest.length();
        return restLength + 1;
    }
    public int totalFrequencies() {
        int restTotalFrequencies = rest.totalFrequencies();
        return restTotalFrequencies + firstWordFrequencyCount;
    }

    public int occurrencesFor(String word) {
        if (firstWord.equals(word))
            return firstWordFrequencyCount;
        else if (firstWord.compareTo(word) < 0)
            return rest.occurrencesFor(word);
        else // Word would have occurred by now if it were present
            return 0;
    }

    public void print() {
        System.out.println(firstWord + " " + firstWordFrequencyCount);
        rest.print();
    }


    public WordFrequencyList capitalize() {
        return new NonEmptyWordFrequencyList(firstWord.toUpperCase(),
                                             firstWordFrequencyCount,
                                             rest.capitalize());
    }

    private String firstWord;               // The first word in the list
    private int firstWordFrequencyCount;    // Its frequency of occurrence
    private WordFrequencyList rest;         // The rest of the list
}
```