HW07 Code

HashTagTokenizer.java:

```
public class HashTagTokenizer {

    public static void main(String[] args) {

        String hashTag = args[0];
        String []dictionary = readDictionary("dictionary.txt");
        breakHashTag(hashTag, dictionary);
    }

    public static String[] readDictionary(String fileName) {
        String[] dictionary = new String[3000];

        In in = new In(fileName);

        for (int i = 0; i < 3000; i++) {
            dictionary[i] = in.readLine();
        }

        return dictionary;
    }

    public static boolean existInDictionary(String word, String
[]dictionary) {
        for (int i = 0; i < dictionary.length; i++) {
            if (dictionary[i].equals(word)) {
                return true;
            }
        }
        return false;
    }

    public static void breakHashTag(String hashtag, String[] dictionary) {
```

```
        // Base case: do nothing (return) if hashtag is an empty string.
        if (hashtag.isEmpty()) {
            return;
        }

        int N = hashtag.length();
        String newString = "";

        for (int i = 1; i <= N; i++) {
            newString = hashtag.substring(0,i);
            if (existInDictionary(newString, dictionary)) {
                System.out.println(newString);
                breakHashTag(hashtag.substring(i), dictionary);
                break;
            }
        }
    }

}
```

SpellChecker.java:

```
public class SpellChecker {


    public static void main(String[] args) {
        String word = args[0];
        int threshold = Integer.parseInt(args[1]);
        String[] dictionary = readDictionary("dictionary.txt");
        String correction = spellChecker(word, threshold, dictionary);
        System.out.println(correction);
    }

    public static String tail(String str) {
        return str.substring(1);
    }

    public static int levenshtein(String word1, String word2) {
        if (word1.length() == 0) {
            return word2.length();
        }else if (word2.length() == 0) {
            return word1.length();
        }else if
(word1.substring(0,1).equalsIgnoreCase(word2.substring(0,1))) {
            return levenshtein(tail(word1), tail(word2));
        }else{

            int minimum = Math.min(levenshtein(tail(word1), word2),
levenshtein(word1, tail(word2)));
            minimum = Math.min(minimum, levenshtein(tail(word1),
tail(word2)));

            return 1+minimum;
        }
    }

    public static String[] readDictionary(String fileName) {
        String[] dictionary = new String[3000];
```

```
        In in = new In(fileName);

        for (int i = 0; i < 3000; i++) {
            dictionary[i] = in.readLine();
        }

        return dictionary;
    }

    public static String spellChecker(String word, int threshold, String[]
dictionary) {
        String newWord = word;
        int limit = threshold + 1;
        for (int i = 0; i < dictionary.length; i++) {
            if (levenshtein (word, dictionary[i]) < limit) {
                limit = levenshtein (word, dictionary[i]);
                newWord = dictionary[i];
            }
        }
        return newWord;
    }

}
```