HashTagTokenizer.java

```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 < 3000; i++){
            if (word.equals(dictionary[i])){
                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();

        for (int i = 1; i <= N; i++) {
            String prefix = hashtag.substring(0, i);

            if (existInDictionary(prefix, dictionary)) {
                System.out.println(prefix);
                breakHashTag(hashtag.substring(i), dictionary);
                return;
            }
```

```
        }
    }

}
```

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) {
        if (str.length() <= 1) {
            return "";
        } else {
            return str.substring(1);
        }
    }

    public static int levenshtein(String word1, String word2) {
        if (word1.isEmpty()) return word2.length();
        if (word2.isEmpty()) return word1.length();
        int diff = 0;
        if (Character.toLowerCase(word1.charAt(0)) ==
Character.toLowerCase(word2.charAt(0))) {
            return levenshtein(tail(word1), tail(word2));
        } else {
            diff++;
            int delete = levenshtein(word1.substring(1), word2) + 1;
            int insert = levenshtein(word1, word2.substring(1)) + 1;
            int replace = levenshtein(word1.substring(1), word2.substring(1)) +
diff;
            return Math.min(Math.min(delete, insert), replace);
        }
    }

    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 closest = word;
        int min = Integer.MAX_VALUE;

        for (int i = 0; i < 3000; i++) {
            String dict = dictionary[i];
            int distance = levenshtein(word.toLowerCase(), dict.toLowerCase());

            if (distance < min) {
                min = distance;
                closest = dict;
            }
        }

        if (min <= threshold) {
            return closest;
        } else {
            return word;
        }
    }

}
```