HW 7 – Neta Tarshish

HashTagTokenizer.java

```
public class HashTagTokenizer {

	public static void main(String[] args) {

		String hashTag = args[0];
		String []dictionary = readDictionary("dictionary.txt");
		breakHashTag(hashTag, dictionary);
		System.out.println(existInDictionary("word",dictionary));
	}

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

		In in = new In(fileName);

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

		return dictionary;
	}

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

```
                return doesExist;
        }

        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 newHashtag = hashtag.toLowerCase();
    String word = "";
    for (int i = 0; i < N; i++) {
        word += newHashtag.charAt(i);
        for(int j = 0; j < dictionary.length; j++){
                if(word.equals(dictionary[j])){
                        System.out.println(word);
                        breakHashTag(newHashtag.substring(i+1,N), 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) {
                return str.substring(1,str.length());
        }

        public static String head(String str) {
                String head = String.valueOf(str.charAt(0));
                return head;
        }

        public static int levenshtein(String word1, String word2) {
  String wordOne = word1.toLowerCase();
  String wordTwo = word2.toLowerCase();
  if (wordOne.equals("")) {
     return wordTwo.length();
  }
  if (wordTwo.equals("")) {
     return wordOne.length();
  }
```

```
    if (head(wordOne).equals(head(wordTwo))) {
        return levenshtein(tail(wordOne), tail(wordTwo));
    }

    int firstCheck = levenshtein(tail(wordOne), wordTwo);
    int secondCheck = levenshtein(wordOne, tail(wordTwo));
    int thirdCheck = levenshtein(tail(wordOne), tail(wordTwo));

    return 1 + Math.min(firstCheck, Math.min(secondCheck, thirdCheck));
}



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

                In in = new In(fileName);

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

                return dictionary;
        }

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

        if (distance <= threshold) {
            if (distance < levenshtein(word, newWord) || newWord.equals(word)) {
```

```
            newWord = dictionary[i];
          }
        }
    }

    return newWord;
}

}
```