```java
/**
 * EECS233 HW6 Programming Project 3
 * Tung Ho Lin
 */

import java.util.Scanner;
import java.io.PrintWriter;
import java.io.IOException;
import java.io.File;

public class WordCounter {

  public WordCounter(){
  }

  public static void main(String args[]) throws IOException {
    String input;
    String output;
    Scanner sc = new Scanner(System.in);
    System.out.println("Path of the input file?");
    input = sc.nextLine();
    System.out.println("Path of the output file?");
    output = sc.nextLine();
    WordCounter.wordCount(input, output);
    sc.close();
  }

  public static String wordCount(String input_file, String output_file) throws IOException {
    String status = "";
    File input = new File(input_file);
    File output = new File(output_file);
    if(input.exists() && output.exists()) {
    MyHashTable words = new MyHashTable();
    Scanner sc = new Scanner(input);
    int wordcount = 0;  //count the total number of words in the inputfile
    while(sc.hasNext()) {
      String word = sc.next();
      wordcount++;
      word = word.toLowerCase();
      //trim the leading punctuations
      word = word.replaceFirst("^[^a-zA-Z]+", "");
      //trim the trailing punctuations
      word = word.replaceAll("[^a-zA-Z]+$", "");
      //omit all white spaces
      word = word.trim();
      //deconstruct the word if there is a punctuation within it
      String[] decon = word.split("\\p{Punct}");
      for(int i=0; i<decon.length; i++) {
        if(words.loadfactor() >= 1)  //if loadfactor is larger than 1, rehash
          words.rehash();  //if the loadfactor of a chaining hashtable is larger than 1, performance will decrease significantly
        words.put(decon[i]); //put the words in the hashtable
      }
    }
    sc.close();
    PrintWriter writer = new PrintWriter(output);
    for(int i=0; i<words.getData().length; i++) {
      if(words.getData()[i] != null) {
        MyHashTable.MyNode curNode = words.getData()[i];
        while(curNode != null) {   //write out all the nodes linked together in each slot of the array
          writer.write("(" + curNode.getData() + " " + curNode.getOccur() + ") ");
          curNode = curNode.getNext();
        }
      }
    }
    writer.close();
    status += "OK; Total words: " + wordcount + ", Hash table size: " + words.getMaxSize();
```

```
67      status += ", Average length of collision lists: " + words.loadfactor();
68      System.out.println(status);
69      }
70      else {
71        status = "File(s) Not Found!";
72        throw new IOException("File(s) Not Found!");
73      }
74      return status;
75    }
76  }
77
```