```java
import java.io.IOException;

/**
 * Takes input file and outputs number of occurrences for each word into an HTML
 * file.
 *
 * @author Gage Farmer
 *
 */
public final class WordCounter {

    /**
     * whitespace.
     *
     */
    private static String[] WHITESPACE = { "-", " ", ".", ",", "!", "?", ";",
            "$", "%" };

    /**
     * No argument constructor--private to prevent instantiation.
     */
    private WordCounter() {
    }

    /**
     * Scans through the input to get all words.
     *
     * @param words
     * @param input
     *
     */
    private static void getWords(SimpleReader input, Queue<String> words) {

        /*
         * Scan through the file and return only space-less lines
         */
        while (!input.atEOS()) {
            String temp = input.nextLine();

            if (!(temp.contains(" ") && temp.isEmpty())) {
                String[] subList = temp.split(" ");

                for (int idx = 0; idx < subList.length; idx++) {
                    String[] cleanedWords = cleanWord(subList[idx]);
                    for (String clean : cleanedWords) {
                        words.enqueue(clean);
                    }

                }
            }

        }
    }

    /**
     * helper method for getWords, just cleans up the list by removing any
     * special characters, and splitting any combined words.
     *
     * @param word
```

```
70      *            words
71      * @return cleaned word
72      */
73     private static String[] cleanWord(String word) {
74
75         // gee i sure hope there isn't an edge case where the array of 10 is too
   small!
76         Boolean t = true;
77         String[] result = null;
78         word = word.toLowerCase();
79
80         for (String special : WHITESPACE) {
81             if (word.contains(special) && t == true) {
82                 result = word.split(special);
83             }
84         }
85         if (result == null) {
86             result = word.split("SUPERDUPERTOPSECRETPHRASE!!!!DON'TTYPEMEEE");
87         }
88
89         return result;
90     }
91
92     /**
93      * Converts two queues to one sorted treemap
94      *
95      * @param word
96      * @param num
97      */
98     private static Map<String, Integer> queueToTreeMap(Queue<String> word) {
99
100        Map<String, Integer> tree = new Map1L<String, Integer>();
101
102        while (word.length() > 0) {
103            String temp = word.dequeue();
104
105            if (!tree.hasKey(temp)) {
106                tree.add(temp, 1);
107            } else {
108                tree.replaceValue(temp, tree.value(temp) + 1);
109            }
110
111        }
112
113        return tree;
114
115    }
116
117    /**
118     * Formats and prints the graph into html.
119     *
120     * @param list
121     * @param output
122     * @throws IOException
123     */
124    private static void printGraph(Map<String, Integer> list,
125            SimpleWriter output) {
126
127        /*
```

```
128          * Intentionally skipping the first key and value (it is whitespace)
129          */
130         Pair key = list.removeAny();
131         int idx = 0;
132 
133         output.println(
134                 "<table style=margin-left:auto;margin-right:auto;> \n <tr> <th>"
135                         + "Word</th> <th>Occurances</th> </tr> \n");
136 
137         while (list.size() > 0) {
138             key = list.removeAny();
139             output.println("<tr> <th>" + key.key() + "</th><th>" + key.value()
140                     + "</th> </tr>\n");
141         }
142 
143     }
144 
145     /**
146      * creates and outputs html header.
147      *
148      * @param output
149      *            file to print to
150      * @param fileName
151      *            name of de file
152      * @throws IOException
153      */
154     private static void printHeader(SimpleWriter output, String fileName) {
155 
156         output.println("<!DOCTYPE html>\n");
157         output.println("<html>\n" + "<style>\n" + "table, th, td {\n"
158                 + "     border:2px solid red;color:#FFFFFF;\n"
159 
160                 + "}\n" + "#grad1 {\n" + "  height: 55px;\n"
161                 + "  background-color: red;\n"
162                 + "  background-image: linear-gradient(to right, red, orange, "
163                 + "yellow, violet);\n" + "}\n" + "</style>\n" + "<head>\n"
164                 + "<div id=\"grad1\" style=\"text-align:center;margin:auto;"
165                 + "color:#FFFFFF;font-size:40px;font-weight:bold\">\n"
166                 + "Words Counted in " + fileName + "\n" + "</div>" + "</head>\n"
167                 + "<body style=\"background-color:#353535;\">\n");
168 
169     }
170 
171     /**
172      * Main method.
173      *
174      * @param args
175      *            the command line arguments
176      * @throws IOException
177      */
178     public static void main(String[] args) {
179         SimpleReader in = new SimpleReader1L();
180         SimpleWriter out = new SimpleWriter1L();
181 
182         /*
183          * Gonna try using queues to keep track of these things
184          */
185         Queue<String> words = new Queue1L<>();
186 
```

```
187         /*
188          * Gets name of file and opens it
189          */
190         out.println("Enter file name: ");
191         String name = in.nextLine();
192         SimpleReader input = new SimpleReader1L(name);
193
194         /*
195          * Gets name of output folder
196          */
197         out.println("Enter the location of the output folder: ");
198         String name2 = in.nextLine();
199
200         /*
201          * Creates html file
202          */
203         SimpleWriter output = new SimpleWriter1L(name2 + "/wordcount.html");
204         printHeader(output, name);
205
206         /*
207          * Gets word data + occurrences to a list
208          */
209         getWords(input, words);
210         Map<String, Integer> list = queueToTreeMap(words);
211
212         /*
213          * Put list in alphabetical order Turn that list into an HTML graph
214          * Print html footer
215          */
216         printGraph(list, output);
217
218         /*
219          * Close input and output streams
220          */
221         in.close();
222         input.close();
223         out.close();
224         output.close();
225
226     }
227
228 }
229
```