```java
import components.map.Map;
import components.map.Map1L;
import components.sequence.Sequence;
import components.sequence.Sequence1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * This program reads a text file, counts occurrences of each word, and
 * generates an HTML document listing words and their counts in alphabetical
 * order using OSU CSE Components.
 *
 * @author David Park
 */
public final class WordCounter {

    /**
     * final string for all non word characters.
     */
    private static final String NON_WORD = "\\W+";

    /**
     * Private constructor to prevent instantiation.
     */
    private WordCounter() {
    }

    /**
     * Reads words from the given file and counts their occurrences.
     *
     * @param inputFilePath
     *            the path to the input file
     * @param wordsMap
     *            the map where word counts are stored
     */
    private static void countWords(String inputFilePath,
            Map<String, Integer> wordsMap) {
        SimpleReader inFile = new SimpleReader1L(inputFilePath);
        while (!inFile.atEOS()) {
            String line = inFile.nextLine();
            Sequence<String> words = parseWords(line);
            // iterate through each word in the sequence
            for (int i = 0; i < words.length(); i++) {
                //set word as the word we receive
                String word = words.entry(i);
                //if the word is in map, we retrieve its count
                if (wordsMap.hasKey(word)) {
                    int count = wordsMap.value(word);
                    //increment and update map
                    wordsMap.replaceValue(word, count + 1);
                } else {
                    wordsMap.add(word, 1);
                }
            }
        }
```

```
58          inFile.close();
59      }
60
61      /**
62       * Parses a line of text into words based on the class-level reg expression.
63       *
64       * @param line
65       *             the line of text to parse
66       * @return a sequence of words
67       */
68      private static Sequence<String> parseWords(String line) {
69          Sequence<String> words = new Sequence1L<>();
70          String[] parts = line.split(NON_WORD);
71          // iterate over each part resulting from the split
72          for (String part : parts) {
73              // check if part is not empty
74              if (!part.isEmpty()) {
75                  // add nonempty part to sequence
76                  words.add(words.length(), part);
77              }
78          }
79          return words;
80      }
81
82      /**
83       * Inserts a word into a sequence in alphabetical order.
84       *
85       * @param sortedWords
86       *             the sequence to insert the word into
87       * @param word
88       *             the word to insert
89       */
90      private static void insertInOrder(Sequence<String> sortedWords,
91              String word) {
92          // initialize position to start at beginning
93          int position = 0;
94          // traverse and find insertion point. continues until it finds a word
95          // alphabetically greater than the other.
96          while (position < sortedWords.length()
97                  && sortedWords.entry(position).compareToIgnoreCase(word) < 0) {
98              position++;
99              // increment the position
100         }
101         // add the new word to the sequence we found
102         sortedWords.add(position, word);
103     }
104
105     /**
106      * Writes the words and their counts to an HTML file, sorted alphabetically.
107      *
108      * @param outputFilePath
109      *             the path to the output HTML file
110      * @param inputFilePath
111      *             the path to the input file, used in the title
112      * @param wordsMap
113      *             the map containing words and their counts
114      */
```

```
115     private static void writeHtml(String outputFilePath, String inputFilePath,
116             Map<String, Integer> wordsMap) {
117         SimpleWriter outFile = new SimpleWriter1L(outputFilePath);
118         // start of HTML Document
119         outFile.println("<!DOCTYPE html>");
120         outFile.println("<html>");
121         outFile.println("<head><title>Word Count</title></head>");
122         outFile.println("<body>");
123
124         // add the heading
125         outFile.println("<h1>Words Counted in " + inputFilePath + "</h1>");
126         outFile.println("<table border=\"1\">");
127         outFile.println("<tr><th>Word</th><th>Count</th></tr>");
128
129         //initalize sequence to hold words sorted
130         Sequence<String> sortedWords = new Sequence1L<>();
131         //iterate through each entry in the map
132         for (Map.Pair<String, Integer> entry : wordsMap) {
133             //insert each word into the sequence in alphabetical order
134             insertInOrder(sortedWords, entry.key());
135         }
136
137         // write each word and its count as a row
138         for (int i = 0; i < sortedWords.length(); i++) {
139             String word = sortedWords.entry(i);
140             outFile.println("<tr><td>" + word + "</td><td>"
141                     + wordsMap.value(word) + "</td></tr>");
142         }
143         // close table and html
144         outFile.println("</table>");
145         outFile.println("</body>");
146         outFile.println("</html>");
147         outFile.close();
148     }
149
150     /**
151      * Main method to run the program.
152      *
153      * @param args
154      *            command line arguments (not used)
155      */
156     public static void main(String[] args) {
157         SimpleReader in = new SimpleReader1L();
158         SimpleWriter out = new SimpleWriter1L();
159
160         // Prompt the user to enter the path to the input file and read the input
161         out.print("Enter the path to the input file: ");
162         String inputFilePath = in.nextLine();
163         // Prompt the user to enter the path where the output HTML file
164         out.print("Enter the path to the output HTML file: ");
165         String outputFilePath = in.nextLine();
166
167         // Initialize a map to store words and their counts
168         Map<String, Integer> wordsMap = new Map1L<>();
169         // Process the input file to count occurrences of each word
170         countWords(inputFilePath, wordsMap);
171         // Generate and write the HTML output
```

```
172         writeHtml(outputFilePath, inputFilePath, wordsMap);
173
174         in.close();
175         out.close();
176     }
177 }
178
```