projects/01-word-counter/src/WordCounter.java

```java
import java.util.Comparator;

import components.map.Map;
import components.map.Map1L;
import components.naturalnumber.NaturalNumber;
import components.naturalnumber.NaturalNumber1L;
import components.queue.Queue;
import components.queue.Queue1L;
import components.set.Set;
import components.set.Set1L;
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;

/**
 * Reads a text file and generates an HTML file with a table displaying each
 * word and its frequency.
 *
 * @author Jared Alonzo
 */
public final class WordCounter {

    /**
     * Private constructor to prevent instantiation.
     */
    private WordCounter() {
        // No initialization needed.
    }

    /**
     * String of separators used to identify words.
     */
    private static final String WORD_SEPARATORS = "., ()-_?/!@#$%^&*\t1234567890:"
            + ";[]{}+=~`><";

    /**
     * Comparator for sorting strings in lexicographical order.
     */
    private static final class StringComparator implements Comparator<String> {
        @Override
        public int compare(String o1, String o2) {
            // Compare two strings lexicographically after converting to lowercase
            return o1.toLowerCase().compareTo(o2.toLowerCase());
        }
    }

    /**
```

```java
     * Outputs closing HTML tags.
     *
     * @param out
     *            the output stream
     */
    private static void outputFooter(SimpleWriter out) {
        // Close the HTML table, body, and html tags
        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }

    /**
     * Outputs words and their frequencies in an HTML table.
     *
     * @param wordList
     *            the list of unique words
     * @param wordOccurrences
     *            a queue containing all occurrences of each word
     * @param out
     *            the output stream
     */
    private static void outputWordAndCount(Queue<String> wordList,
            Queue<String> wordOccurrences, SimpleWriter out) {
        // Create a map to store word counts
        Map<String, NaturalNumber> wordCountMap = new Map1L<>();

        // Initialize the map with each word having a count of 0
        for (String word : wordList) {
            wordCountMap.add(word, new NaturalNumber1L());
        }

        // Count the occurrences of each word
        for (String word : wordOccurrences) {
            if (wordCountMap.hasKey(word)) {
                wordCountMap.value(word).increment();
            }
        }

        // Output each word and its count in an HTML table row
        for (String word : wordList) {
            out.println("<tr><td>" + word + "</td><td>"
                    + wordCountMap.value(word) + "</td></tr>");
        }
    }

    /**
     * Extracts the next word or separator string from the given text.
     *
     * @param text
     *            the input text
     * @param position
     *            the starting position
     * @param separators
     *            the set of separator characters
     * @return the next word or separator string
     */
    private static String nextWordOrSeparator(String text, int position,
            Set<Character> separators) {
        assert text != null : "Violation of: text is not null";
        assert separators != null : "Violation of: separators is not null";
        assert 0 <= position : "Violation of: 0 <= position";
        assert position < text.length() : "Violation of: position < |text|";

        // Initialize the first character and the result string
        char firstChar = text.charAt(position);
        StringBuilder result = new StringBuilder();
        int i = position;

        // Determine if the first character is a separator
        boolean isSeparator = separators.contains(firstChar);

        // Extract the next word or separator string
        while (i < text.length()
                && separators.contains(text.charAt(i)) == isSeparator) {
            result.append(text.charAt(i));
            i++;
        }

        return result.toString();
    }

    /**
     * Generates a set of unique characters from the given string.
     *
     * @param str
     *            the input string
     * @param strSet
     *            the set to be filled with characters from the string
     */
    private static void generateElements(String str, Set<Character> strSet) {
        assert str != null : "Violation of: str is not null";
        assert strSet != null : "Violation of: strSet is not null";

        // Add each character in the string to the set
        for (int i = 0; i < str.length(); i++) {
            strSet.add(str.charAt(i));
        }
    }

    /**
     * Populates word lists with words from the input file and sorts the unique
     * words alphabetically.
     *
     * @param wordList
     *            the list of unique words
     * @param wordOccurrences
     *            the list of all word occurrences
     * @param fileData
     *            the input stream containing the text file
     */
    private static void getList(Queue<String> wordList,
            Queue<String> wordOccurrences, SimpleReader fileData) {
        // Create a set of separator characters
        Set<Character> separators = new Set1L<>();
        generateElements(WORD_SEPARATORS, separators);

        // Queue to store each line from the file
        Queue<String> linesFromFile = new Queue1L<>();
        while (!fileData.atEOS()) {
            linesFromFile.enqueue(fileData.nextLine());
        }

        // Process each line to extract words
        while (linesFromFile.length() > 0) {
            String line = linesFromFile.dequeue();
            int position = 0;

            // Extract each word or separator string from the line
            while (position < line.length()) {
                String word = nextWordOrSeparator(line, position, separators);
                position += word.length();

                // If the word is not a separator, add it to the lists
                if (!separators.contains(word.charAt(0))) {
                    boolean containsWord = false;
                    for (String w : wordList) {
                        containsWord = containsWord || w.equals(word);
                    }
                    if (!containsWord) {
                        wordList.enqueue(word);
                    }
                    wordOccurrences.enqueue(word);
                }
            }
        }

        // Sort the list of unique words alphabetically
        wordList.sort(new StringComparator());
    }

    /**
     * Outputs the opening HTML tags and table headers.
     *
     * @param fileOut
     *            the output stream
     * @param userInput
     *            the title of the HTML file
     */
    private static void outputHeader(SimpleWriter fileOut, String userInput) {
        // Generate the opening HTML structure and table headers
        fileOut.println("<html>");
        fileOut.println("<style>");
        fileOut.println("table, th, td { border:1px solid black; }");
        fileOut.println("</style>");
        fileOut.println("<head><title>Words Counted in " + userInput
                + "</title></head>");
        fileOut.println("<body>");
        fileOut.println("<h3>Words Counted in " + userInput + "</h3>");
        fileOut.println("<hr class=\"new1\">");
        fileOut.println(
                "<table style=\"width:10%\"><tr><th>Words</th><th>Counts</th></tr>");
    }

    /**
     * Processes the input file and generates an HTML file with word counts.
     *
     * @param userInput
     *            the name of the input file
     * @param outputFile
     *            the name of the output HTML file
     */
    private static void processFile(String userInput, String outputFile) {
        // Create streams for reading the input file and writing the output file
        SimpleWriter fileOut = new SimpleWriter1L(outputFile);
        SimpleReader fileData = new SimpleReader1L(userInput);

        // Queues to hold unique words and all word occurrences
        Queue<String> wordList = new Queue1L<>();
        Queue<String> wordOccurrences = new Queue1L<>();

        // Generate the HTML header and process the file content
        outputHeader(fileOut, userInput);
        getList(wordList, wordOccurrences, fileData);
        outputWordAndCount(wordList, wordOccurrences, fileOut);
        outputFooter(fileOut);

        // Close the file streams
        fileOut.close();
        fileData.close();
    }

    /**
     * Main method.
     *
     * @param args
     *            command line arguments; unused
     */
    public static void main(String[] args) {
        // Create streams for user input and output
        SimpleWriter out = new SimpleWriter1L();
        SimpleReader in = new SimpleReader1L();

        // Prompt the user for the input file name and output file name
        out.print("Enter the name of the input file: ");
        String userInput = in.nextLine();
        out.print("Enter the name of the output file: ");
        String outputFile = in.nextLine();

        // Process the file and generate the HTML output
        processFile(userInput, outputFile);

        // Notify the user of success and close the streams
        out.println("Success!");
        out.close();
        in.close();
    }
}

```