```java
import java.io.File;

/**
 * Creates a glossary of terms for a given input
 *
 * @author Gage Farmer
 *
 */
public final class Glossary {

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

    /**
     * Takes term and its definition from the in-file, and formats it properly
     * into the out-file.
     *
     * @param in
     *            input file
     * @param out
     *            output file
     */
    private static void generateTerm(TreeMap<String, String> in,
            SimpleWriter out) {

        String fileName, definition;
        out.println("<ul>");

        for (Entry<String, String> entry : in.entrySet()) {
            String term = entry.getKey();
            if (!term.equals("")) {
                out.println("<li><a href=\"pages/" + term + ".html\">" + term
                        + "</a></li>");
                fileName = "data/pages/" + term + ".html";
                definition = entry.getValue();
                generateDefinition(term, definition, fileName);
            }

        }

        out.println("</ul>");
    }

    /**
     * Takes term and its definition from the in-file, and formats it properly
     * into the out-file.
     *
     * @param in
     *            input file
     * @param out
     *            output file
     */
    private static void generateDefinition(String term, String definition,
            String fileName) {

        File file = new File(fileName);
```

```
71          try {
72              file.createNewFile();
73          } catch (IOException e) {
74              // TODO Auto-generated catch block
75              e.printStackTrace();
76          }
77
78          SimpleWriter out = new SimpleWriter1L(fileName);
79
80          out.println("<i><h1><b style='color:red;'>" + term
81                  + "</b></i></h1><br><head>" + definition + "</head>");
82          out.println(
83                  "<br><br>Return to <a href=\"javascript:history.back()\">index</a>");
84
85          out.close();
86      }
87
88      /**
89       * turn file into a map.
90       *
91       * @param in
92       *            infile
93       * @param length
94       *            length of file
95       * @return map!!!!
96       */
97      private static TreeMap<String, String> generateMap(SimpleReader in,
98              int length) {
99          int line = 1, i = 0;
100         String next = in.nextLine(), term = "", definition = "";
101         TreeMap<String, String> map = new TreeMap<>();
102
103         while (line < length) {
104             if (!next.equals("")) {
105
106                 if (i == 0) {
107                     term = next;
108                 } else {
109                     definition = definition + " " + next;
110                 }
111                 next = in.nextLine();
112                 line++;
113                 i++;
114
115             } else {
116                 map.put(term, definition);
117                 term = "";
118                 definition = "";
119                 i = 0;
120                 next = in.nextLine();
121                 line++;
122             }
123
124         }
125         map.put(term, definition); // brute force get fcked java
126
127         return map;
128
129     }
```

```
130
131     /**
132      * Generates the header for the html file.
133      *
134      * @param out
135      *            outfile!!!!
136      */
137     public static void generateHeader(SimpleWriter out) {
138         out.println("<?xml version=\"1.0\" encoding=\"ISO-8859-1\" ?>");
139         out.println(
140                 "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\" \"h"
141                         + "ttp://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">");
142         out.println("<html xmlns=\"http://www.w3.org/1999/xhtml\">");
143         out.println("<h1>Glossary<h1>");
144         out.println("<h3>Index</h3>");
145     }
146
147     /**
148      * counts the lines in the file. I'll be honest, i ripped this code off the
149      * Internet but honestly it felt like a huge waste of time to try and figure
150      * it out on my own. i don't feel sorry at all. this code wasn't the main
151      * focus of the project. why would i spend my time counting lines when i can
152      * spend time on more important things like Counter-Strike: Global
153      * Offensive?? Did you know I'm top 2 in the world?? I'm a GAMER!!! I
154      * GAMEE!!!!! YOU CANT HOLD ME DOWN WITH YOUR SILLY LINE COUNTING
155      * SHENANIGANS!!!!
156      * IIIIIIIIIIIIIAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA!!!!!!!!!!
157      *
158      * @param fileName
159      *            name of file LOL!!!!
160      * @return the number of lines LOL!!!!!
161      * @throws IOException
162      */
163     public static long countLines(String fileName) throws IOException {
164
165         Path path = Paths.get(fileName);
166
167         long lines = 0;
168         try {
169
170             lines = Files.lines(path).count();
171
172         } catch (IOException e) {
173             e.printStackTrace();
174         }
175
176         return lines;
177
178     }
179
180     /**
181      * Main method.
182      *
183      * @param args
184      *            the command line arguments
185      * @throws IOException
186      */
187     public static void main(String[] args) throws IOException {
188         SimpleReader getName = new SimpleReader1L();
```

```
189         SimpleWriter nameGet = new SimpleWriter1L();
190
191         nameGet.print("Enter a file name: ");
192         String fileName = getName.nextLine();
193
194         // getting file name and stuff
195         if (!fileName.substring(0, 5).equals("data/")) {
196             fileName = "data/" + fileName;
197         }
198         if (!fileName.substring(fileName.length() - 4, fileName.length() - 3)
199                 .equals(".")) {
200             fileName = fileName + ".txt";
201         }
202
203         getName.close();
204         nameGet.close();
205
206         // i hate comments
207         int fileLength = (int) countLines(fileName);
208         SimpleReader in = new SimpleReader1L(fileName);
209         SimpleWriter out = new SimpleWriter1L("data/glossary.html");
210         TreeMap<String, String> map;
211
212         map = generateMap(in, fileLength);
213         generateHeader(out);
214         generateTerm(map, out);
215
216         in.close();
217         out.close();
218     }
219
220 }
221
```