```java
import components.simplereader.SimpleReader;

/**
 * Program to convert an XML RSS (version 2.0) feed from a given URL into the
 * corresponding HTML output file.
 *
 * @author David Park
 *
 */
public final class RSSAggregator {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private RSSAggregator() {
    }

    /**
     * Outputs the "opening" tags in the generated HTML file. These are the
     * expected elements generated by this method:
     *
     * <html> <head> <title>the channel tag title as the page title</title>
     * </head> <body>
     * <h1>the page title inside a link to the <channel> link</h1>
     * <p>
     * the channel description
     * </p>
     * <table border="1">
     * <tr>
     * <th>Date</th>
     * <th>Source</th>
     * <th>News</th>
     * </tr>
     *
     * @param channel
     *            the channel element XMLTree
     * @param out
     *            the output stream
     * @updates out.content
     * @requires [the root of channel is a <channel> tag] and out.is_open
     * @ensures out.content = #out.content * [the HTML "opening" tags]
     */
    private static void outputHeader(XMLTree channel, SimpleWriter out) {
        assert channel != null : "Violation of: channel is not null";
        assert out != null : "Violation of: out is not null";
        assert channel.isTag() && channel.label().equals("channel") : ""
                + "Violation of: the label root of channel is a <channel> tag";
        assert out.isOpen() : "Violation of: out.is_open";

        // get the channel title
        int titleIndex = getChildElement(channel, "title");
        String title = "Empty Title"; // set as default
        if (titleIndex >= 0
                && channel.child(titleIndex).numberOfChildren() > 0) {
            title = channel.child(titleIndex).child(0).label();
        }

```

```
63          // extract the channel link
64          int linkIndex = getChildElement(channel, "link");
65          String link = ""; // set as default
66
67          link = channel.child(linkIndex).child(0).label();
68
69          //declare descriptionIndex with index of description
70          int descriptionIndex = getChildElement(channel, "description");
71          String description = "";
72          //check if description exists and has atleast one child
73          if (descriptionIndex >= 0
74                  && channel.child(descriptionIndex).numberOfChildren() > 0) {
75              description = channel.child(descriptionIndex).child(0).label();
76          }
77
78          //print out rss/html info
79          out.println("<html>");
80          out.println("<head>");
81          out.println("<title>" + title + "</title>");
82          out.println("</head>");
83          out.println("<body>");
84          out.println("<h1><a href=\"" + link + "\">" + title + "</a></h1>");
85          out.println("<p>" + description + "</p>");
86          out.println("<table border=\"1\">");
87          out.println("<tr>");
88          out.println("<th>Date</th>");
89          out.println("<th>Source</th>");
90          out.println("<th>News</th>");
91          out.println("</tr>");
92      }
93
94      /**
95       * Outputs the "closing" tags in the generated HTML file. These are the
96       * expected elements generated by this method:
97       *
98       * </table>
99       * </body> </html>
100      *
101      * @param out
102      *            the output stream
103      * @updates out.contents
104      * @requires out.is_open
105      * @ensures out.content = #out.content * [the HTML "closing" tags]
106      */
107     private static void outputFooter(SimpleWriter out) {
108         assert out != null : "Violation of: out is not null";
109         assert out.isOpen() : "Violation of: out.is_open";
110
111         // Output the closing HTML tags
112         out.println("</table>");
113         out.println("</body>");
114         out.println("</html>");
115     }
116
117     /**
118      * Finds the first occurrence of the given tag among the children of the
119      * given {@code XMLTree} and return its index; returns -1 if not found.
```

```
120      *
121      * @param xml
122      *            the {@code XMLTree} to search
123      * @param tag
124      *            the tag to look for
125      * @return the index of the first child of type tag of the {@code XMLTree}
126      *         or -1 if not found
127      * @requires [the label of the root of xml is a tag]
128      * @ensures <pre>
129      * getChildElement =
130      *  [the index of the first child of type tag of the {@code XMLTree} or
131      *   -1 if not found]
132      * </pre>
133      */
134     private static int getChildElement(XMLTree xml, String tag) {
135         assert xml != null : "Violation of: xml is not null";
136         assert tag != null : "Violation of: tag is not null";
137         assert xml.isTag() : "Violation of: the label root of xml is a tag";
138
139         // initialize as if the tag is not found.
140         int result = -1;
141
142         // iterate through children of XMLTree
143         for (int i = xml.numberOfChildren() - 1; i >= 0; i--) {
144             if (xml.child(i).label().equals(tag)) {
145                 // if child is found with matching tag, update result with its index.
146                 result = i;
147             }
148         }
149         return result;
150     }
151
152     /**
153      * Processes one news item and outputs one table row. The row contains three
154      * elements: the publication date, the source, and the title (or
155      * description) of the item.
156      *
157      * @param item
158      *            the news item
159      * @param out
160      *            the output stream
161      * @updates out.content
162      * @requires [the label of the root of item is an <item> tag] and
163      *           out.is_open
164      * @ensures <pre>
165      * out.content = #out.content *
166      *   [an HTML table row with publication date, source, and title of news item]
167      * </pre>
168      */
169     private static void processItem(XMLTree item, SimpleWriter out) {
170         assert item != null : "Violation of: item is not null";
171         assert out != null : "Violation of: out is not null";
172         assert item.isTag() && item.label().equals(
173                 "item") : "Violation of: the label root of item is an <item> tag";
174         assert out.isOpen() : "Violation of: out.is_open";
175
176         String date = "No Date Available";
```

```
177         String source = "No Source Available";
178         String titleOrDescription = "No Title Available";
179         String link = "";
180 
181         // Extract date
182         int dateIndex = getChildElement(item, "pubDate");
183         if (dateIndex >= 0 && item.child(dateIndex).numberOfChildren() > 0) {
184             date = item.child(dateIndex).child(0).label();
185         }
186 
187         // Extract source
188         int sourceIndex = getChildElement(item, "source");
189         if (sourceIndex >= 0 && item.child(sourceIndex).hasAttribute("url")) {
190             String sourceUrl = item.child(sourceIndex).attributeValue("url");
191             if (item.child(sourceIndex).numberOfChildren() > 0) {
192                 String sourceText = item.child(sourceIndex).child(0).label();
193                 source = "<a href=\"" + sourceUrl + "\">" + sourceText + "</a>";
194             } else {
195                 source = "<a href=\"" + sourceUrl + "\">" + sourceUrl + "</a>";
196             }
197         }
198 
199         // Extract title
200         int titleIndex = getChildElement(item, "title");
201         int descriptionIndex = getChildElement(item, "description");
202         if (titleIndex >= 0 && item.child(titleIndex).numberOfChildren() > 0) {
203             titleOrDescription = item.child(titleIndex).child(0).label();
204         } else if (descriptionIndex >= 0
205                 && item.child(titleIndex).numberOfChildren() > 0) {
206             titleOrDescription = item.child(titleIndex).child(0).label();
207         }
208 
209         // Check if a link is available and embed it in the title if so
210         int linkIndex = getChildElement(item, "link");
211         if (linkIndex >= 0 && item.child(linkIndex).numberOfChildren() > 0) {
212             link = item.child(linkIndex).child(0).label();
213             // Embed the link in the title
214             titleOrDescription = "<a href=\"" + link + "\">"
215                     + titleOrDescription + "</a>";
216         }
217 
218         // Output the information
219         out.println("<tr>");
220         out.println("<td>" + date + "</td>");
221         out.println("<td>" + source + "</td>");
222         out.println("<td>" + titleOrDescription + "</td>");
223         out.println("</tr>");
224     }
225 
226     /**
227      * Processes one XML RSS (version 2.0) feed from a given URL converting it
228      * into the corresponding HTML output file.
229      *
230      * @param url
231      *            the URL of the RSS feed
232      * @param file
233      *            the name of the HTML output file
```

```
234      * @param out
235      *            the output stream to report progress or errors
236      * @updates out.content
237      * @requires out.is_open
238      * @ensures <pre>
239      * [reads RSS feed from url, saves HTML document with table of news items
240      *   to file, appends to out.content any needed messages]
241      * </pre>
242      */
243     private static void processFeed(String url, String file, SimpleWriter out) {
244         XMLTree xml = new XMLTree1(url);
245         // Check if the XML document is a valid RSS 2.0 feed
246         if (xml.label().equals("rss") && xml.hasAttribute("version")
247                 && xml.attributeValue("version").equals("2.0")) {
248             SimpleWriter fileOut = new SimpleWriter1L(file);
249             XMLTree channel = xml.child(0);
250             outputHeader(channel, fileOut);
251 
252             // Iterate through all children of channel
253             for (int i = 0; i < channel.numberOfChildren(); i++) {
254                 // Check if label is item
255                 if (channel.child(i).label().equals("item")) {
256                     // Process item element with processItem
257                     processItem(channel.child(i), fileOut);
258                 }
259             }
260 
261             outputFooter(fileOut);
262             fileOut.close();
263             out.println("Processed feed: " + url + " into " + file);
264         } else {
265             out.println("URL is not a valid RSS 2.0 Feed: " + url);
266         }
267     }
268 
269     /**
270      * Main method.
271      *
272      * @param args
273      *            the command line arguments; unused here
274      */
275     public static void main(String[] args) {
276         SimpleReader in = new SimpleReader1L();
277         SimpleWriter out = new SimpleWriter1L();
278 
279         out.println(
280                 "Enter the name of the XML file containing the RSS feed URLs:");
281         String inputFile = in.nextLine();
282         XMLTree feedList = new XMLTree1(inputFile);
283 
284         //ask user for name of output
285         out.println("Enter name of the index HTML output file:");
286         //read name of output file
287         String indexFileName = in.nextLine();
288 
289         //declare a writer to write HTML output to file.
290         SimpleWriter indexFileOut = new SimpleWriter1L(indexFileName);
```

```
291         // write basic structure of html
292         indexFileOut.println("<html>");
293         indexFileOut.println("<head>");
294         indexFileOut.println(
295                 "<title>" + feedList.attributeValue("title") + "</title>");
296         indexFileOut.println("</head>");
297         indexFileOut.println("<body>");
298         indexFileOut
299                 .println("<h1>" + feedList.attributeValue("title") + "</h1>");
300         indexFileOut.println("<ul>");
301         // create the list for the HTML
302 
303         for (int i = 0; i < feedList.numberOfChildren(); i++) {
304             //get the current feed for the XMLTree.
305             XMLTree feed = feedList.child(i);
306             // extract url, name and file attributes for the current feed
307             String url = feed.attributeValue("url");
308             String name = feed.attributeValue("name");
309             String file = feed.attributeValue("file");
310 
311             processFeed(url, file, out); // Process each feed
312 
313             indexFileOut.println(
314                     "<li><a href=\"" + file + "\">" + name + "</a></li>");
315         }
316 
317         //close the list, body and html tags to format the HTML
318         indexFileOut.println("</ul>");
319         indexFileOut.println("</body>");
320         indexFileOut.println("</html>");
321 
322         indexFileOut.close();
323         in.close();
324         out.close();
325     }
326 }
327 
```