```
 1 import components.simplereader.SimpleReader;
 2 import components.simplereader.SimpleReader1L;
 3 import components.simplewriter.SimpleWriter;
 4 import components.simplewriter.SimpleWriter1L;
 5 import components.xmltree.XMLTree;
 6 import components.xmltree.XMLTree1;
 7
 8 /**
 9  * Program to convert an XML RSS (version 2.0) feed from a
   given URL into the
10  * corresponding HTML output file.
11  *
12  * @author Ansh Pachauri
13  *
14  */
15 public final class RSSReader {
16
17     /**
18      * Private constructor so this utility class cannot be
   instantiated.
19      */
20     private RSSReader() {
21     }
22
23     /**
24      * Outputs the "opening" tags in the generated HTML file.
   These are the
25      * expected elements generated by this method:
26      *
27      * <html> <head> <title>the channel tag title as the page
   title</title>
28      * </head> <body>
29      * <h1>the page title inside a link to the <channel> link</
   h1>
30      * <p>
31      * the channel description
32      * </p>
33      * <table border="1">
34      * <tr>
```

```
35      * <th>Date</th>
36      * <th>Source</th>
37      * <th>News</th>
38      * </tr>
39      *
40      * @param channel
41      *            the channel element XMLTree
42      * @param out
43      *            the output stream
44      * @updates out.content
45      * @requires [the root of channel is a <channel> tag] and
    out.is_open
46      * @ensures out.content = #out.content * [the HTML
    "opening" tags]
47      */
48     private static void outputHeader(XMLTree channel,
    SimpleWriter out) {
49         /*
50          * If title has child then the text will be assigned to
    String title,
51          * otherwise, the String title will output "Empty
    Title".
52          */
53         String title;
54         if (getChildElement(channel, "title") >= 0) {
55             if (channel.child(getChildElement(channel,
    "title"))
56                     .numberOfChildren() >= 1) {
57                 title = channel.child(getChildElement(channel,
    "title"))
58                         .child(0).label();
59             } else {
60                 title = "Empty Title";
61             }
62         } else if (getChildElement(channel, "description") >=
    0) {
63             if (channel.child(getChildElement(channel,
    "description"))
64                     .numberOfChildren() >= 1) {
```

```
65                 title = channel.child(getChildElement(channel,
   "description"))
66                             .label();
67             } else {
68                 title = "Empty Title";
69             }
70
71         } else {
72             title = "Empty Title";
73         }
74
75         /*
76          * If description has child then the text will be
   assigned to String
77          * desc, otherwise, the String desc will output "No
   description".
78          */
79         String desc = "";
80         if (channel.child(getChildElement(channel,
   "description"))
81                 .numberOfChildren() >= 0) {
82             if (channel.child(getChildElement(channel,
   "description"))
83                     .numberOfChildren() >= 1) {
84                 desc = channel.child(getChildElement(channel,
   "description"))
85                             .child(0).label();
86             }
87
88         } else {
89             desc = "No Description.";
90         }
91         // header
92         out.println("<html>");
93         out.println("<head>");
94         out.println("<title>" + title + "</title>");
95         out.println("</head>");
96         out.println("<body>");
97         out.println(
```

```
 98                 "<h1><a href=\""
 99                         +
   channel.child(getChildElement(channel, "link"))
100                                 .child(0).label()
101                         + "\">" + title + "</a></h1>");
102         out.println("<p>" + desc + "</p>");
103         out.println("<table border=\"1\">");
104         out.println("<tr>");
105         out.println("<th>Date</th>");
106         out.println("<th>Source</th>");
107         out.println("<th>News</th>");
108         out.println("</tr>");
109
110     }
111
112     /**
113      * Outputs the "closing" tags in the generated HTML file.
   These are the
114      * expected elements generated by this method:
115      *
116      * </table>
117      * </body> </html>
118      *
119      * @param out
120      *             the output stream
121      * @updates out.contents
122      * @requires out.is_open
123      * @ensures out.content = #out.content * [the HTML
   "closing" tags]
124      */
125     private static void outputFooter(SimpleWriter out) {
126
127         out.println("</table>");
128         out.println("</body>");
129         out.println("</html>");
130     }
131
132     /**
133      * Finds the first occurrence of the given tag among the
```

```
    children of the
134      * given {@code XMLTree} and return its index; returns -1
    if not found.
135      *
136      * @param xml
137      *            the {@code XMLTree} to search
138      * @param tag
139      *            the tag to look for
140      * @return the index of the first child of type tag of the
    {@code XMLTree}
141      *         or -1 if not found
142      * @requires [the label of the root of xml is a tag]
143      * @ensures <pre>
144      * getChildElement =
145      *  [the index of the first child of type tag of the {@code
    XMLTree} or
146      *   -1 if not found]
147      * </pre>
148      */
149     private static int getChildElement(XMLTree xml, String tag)
    {
150
151         int index = -1;
152         int i = 0;
153         while (i < xml.numberOfChildren() && index == -1) {
154
155             if (xml.child(i).label().equals(tag)) {
156                 index = i;
157             }
158         }
159         return index;
160     }
161
162     /**
163      * Processes one news item and outputs one table row. The
    row contains three
164      * elements: the publication date, the source, and the
    title (or
165      * description) of the item.
```

```
166      *
167      * @param item
168      *             the news item
169      * @param out
170      *             the output stream
171      * @updates out.content
172      * @requires [the label of the root of item is an <item>
    tag] and
173      *           out.is_open
174      * @ensures <pre>
175      * out.content = #out.content *
176      *   [an HTML table row with publication date, source, and
    title of news item]
177      * </pre>
178      */
179     private static void processItem(XMLTree item, SimpleWriter
    out) {
180         //table start
181         out.println("<tr>");
182         //assigns pubDate with the date then prints the row
    item
183         String pubDate;
184         if (getChildElement(item, "pubDate") >= 0) {
185             if (item.child(getChildElement(item, "pubDate"))
186                     .numberOfChildren() > 0) {
187                 pubDate = item.child(getChildElement(item,
    "pubDate")).child(0)
188                         .label();
189             } else {
190                 pubDate = "No date available";
191             }
192
193         } else {
194             pubDate = "No date available";
195         }
196         out.println("<th>" + pubDate + "</th>");
197         //assigns source with the source link then prints the
    row item
198         String source;
```

```
199         String sourceLink = "";
200         int i = 0;
201         if (getChildElement(item, "source") >= 0) {
202             if (item.child(getChildElement(item, "source"))
203                     .numberOfChildren() > 0) {
204                 source = item.child(getChildElement(item,
    "source")).child(0)
205                         .label();
206                 if (item.child(getChildElement(item, "source"))
207                         .hasAttribute("url")) {
208                     sourceLink =
    item.child(getChildElement(item, "source"))
209                             .attributeValue("url");
210                     i = 1;
211                 } else {
212                     i = 0;
213                 }
214 
215             } else {
216                 source = "No source available";
217             }
218 
219         } else {
220             source = "No source available";
221         }
222 
223         if (i == 1) {
224             out.println(
225                     "<th><a href=\"" + sourceLink + "\">" +
    source + "</th>");
226         } else {
227             out.println("<th>" + source + "</th>");
228         }
229         //assigns title with the title of the article then
    prints the row item
230         String title = "";
231         if (getChildElement(item, "title") >= 0) {
232             if (item.child(getChildElement(item, "title"))
233                     .numberOfChildren() >= 1) {
```

```
234                 title = item.child(getChildElement(item,
    "title")).child(0)
235                         .label();
236             }
237
238         } else if (getChildElement(item, "description") >= 0) {
239             if (item.child(getChildElement(item,
    "description"))
240                     .numberOfChildren() >= 1) {
241                 title = item.child(getChildElement(item,
    "description"))
242                         .child(0).label();
243             }
244         } else {
245             title = "No title available";
246         }
247
248         String link = "";
249         if (getChildElement(item, "link") >= 0) {
250             if (item.child(getChildElement(item, "link"))
251                     .numberOfChildren() >= 1) {
252                 link = item.child(getChildElement(item,
    "link")).child(0)
253                         .label();
254             }
255         }
256
257         out.println("<th><a href=\"" + link + "\">" + title +
    "</th>");
258
259         out.println("</tr>");
260
261     }
262
263     /**
264      * Main method.
265      *
266      * @param args
267      *             the command line arguments; unused here
```

```
268      */
269     public static void main(String[] args) {
270         SimpleReader in = new SimpleReader1L();
271         SimpleWriter out = new SimpleWriter1L();
272 
273         out.print("Enter the URL of an RSS 2.0 news feed: ");
274         String url = in.nextLine();
275         XMLTree xml = new XMLTree1(url);
276 
277         out.print("Enter output file name: ");
278         String outFileName = in.nextLine();
279         //checking if the url provided is a valid RSS 2.0 feed
    url
280         while (!xml.label().equals("rss") && !
    xml.hasAttribute("version")
281                 && !
    xml.attributeValue("version").equals("2.0")) {
282             out.print("\nEnter a valid URL of a RSS 2.0 feed:
    ");
283             url = in.nextLine();
284             xml = new XMLTree1(url);
285         }
286         SimpleWriter fileOut = new SimpleWriter1L(outFileName);
287         outputHeader(xml.child(0), fileOut);
288         int i = 0;
289         while (xml.child(0).numberOfChildren() > i) {
290             if (xml.child(0).child(i).label().equals("item")) {
291                 processItem(xml.child(0).child(i), fileOut);
292             }
293             i++;
294         }
295         outputFooter(fileOut);
296 
297         in.close();
298         out.close();
299     }
300 
301 }
302 
```