```java
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
import components.xmltree.XMLTree;
import components.xmltree.XMLTree1;

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

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

    /**
     * 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";

        // title
        int titleNum = getChildElement(channel, "title");
        XMLTree title = channel.child(titleNum);

        out.print("<html> <head> <title>");
        if (title.numberOfChildren() > 0) {
            out.print(title.child(0).label());
        }
        out.println("</title>");

        out.println("</head> <body>");

        //link
        int linkNum = getChildElement(channel, "link");
        XMLTree link = channel.child(linkNum);

        out.print("<h1>");
        if (link.numberOfChildren() > 0) {
            out.print("<a href=\"" + link.child(0).label() + ">");
            if (title.numberOfChildren() > 0) {
                out.print(title.child(0).label());
            }
            out.println("</a></h1>");
        }

        //description
        int descriptionNum = getChildElement(channel, "description");
        XMLTree description = channel.child(descriptionNum);

        out.print("<p>");
        if (description.numberOfChildren() > 0) {
            out.print(description.child(0).label());
        }
        out.println("</p>");

        out.println("<table border=\"1\">");
        out.println("<tr><th>Date</th><th>Source</th><th>News</th></tr>");
    }

    /**
     * Outputs the "closing" tags in the generated HTML file. These are the
     * expected elements generated by this method:
     *
     * </table>
     * </body> </html>
     *
     * @param out
     *            the output stream
     * @updates out.contents
     * @requires out.is_open
     * @ensures out.content = #out.content * [the HTML "closing" tags]
     */
    private static void outputFooter(SimpleWriter out) {
        assert out != null : "Violation of: out is not null";
        assert out.isOpen() : "Violation of: out.is_open";

        out.println("</table>");
        out.println("</body>");
        out.println("</html>");
    }

    /**
     * Finds the first occurrence of the given tag among the children of the
     * given {@code XMLTree} and return its index; returns -1 if not found.
     *
     * @param xml
     *            the {@code XMLTree} to search
     * @param tag
     *            the tag to look for
     * @return the index of the first child of type tag of the {@code XMLTree}
     *         or -1 if not found
     * @requires [the label of the root of xml is a tag]
     * @ensures <pre>
     * getChildElement =
     *  [the index of the first child of type tag of the {@code XMLTree} or
     *   -1 if not found]
     * </pre>
     */
    private static int getChildElement(XMLTree xml, String tag) {
        assert xml != null : "Violation of: xml is not null";
        assert tag != null : "Violation of: tag is not null";
        assert xml.isTag() : "Violation of: the label root of xml is a tag";

        int n = xml.numberOfChildren();
        int index = -1;
        int i = 0;

        while (i < n && index == -1) {
            if (xml.child(i).label().equals(tag)) {
                index = i;
            }
            i++;
        }
        return index;
    }

    /**
     * Processes one news item and outputs one table row. The row contains three
     * elements: the publication date, the source, and the title (or
     * description) of the item.
     *
     * @param item
     *            the news item
     * @param out
     *            the output stream
     * @updates out.content
     * @requires [the label of the root of item is an <item> tag] and
     *           out.is_open
     * @ensures <pre>
     * out.content = #out.content *
     *   [an HTML table row with publication date, source, and title of news item]
     * </pre>
     */
    private static void processItem(XMLTree item, SimpleWriter out) {
        assert item != null : "Violation of: item is not null";
        assert out != null : "Violation of: out is not null";
        assert item.isTag() && item.label().equals("item") : ""
```

```
                + "Violation of: the label root of item is an <item> tag";
        assert out.isOpen() : "Violation of: out.is_open";

        int indexPubDate = getChildElement(item, "pubDate");
        int indexSource = getChildElement(item, "source");
        int indexTitle = getChildElement(item, "title");
        int indexDescription = getChildElement(item, "description");
        int indexLink = getChildElement(item, "link");

        out.println("<tr>");

        //publication date (if pubDate exists, it is required to have a child)
        if (indexPubDate >= 0) {
            out.println("<td>" + item.child(indexPubDate).child(0).label()
                    + "</td>");
        } else {
            out.println("<td>No date available</td>");
        }

        //source
        if (indexSource >= 0
                && item.child(indexSource).numberOfChildren() > 0) {
            XMLTree source = item.child(indexSource);
            out.print("<td>");
            out.print("<a href=\"" + source.attributeValue("url") + "\">");
            out.println(source.child(0).label() + "</a></td>");
        } else {
            out.println("<td>No source available</td>");
        }

        // link (if link exists, it is required to have a child)

        if (indexTitle >= 0 && item.child(indexTitle).numberOfChildren() > 0) {
            XMLTree title = item.child(indexTitle);
            out.print("<td>");
            if (indexLink >= 0
                    && item.child(indexLink).numberOfChildren() > 0) {
                XMLTree link = item.child(indexLink);
                out.print("<a href=\"" + link.child(0).label() + "\">");
            }
            out.println(title.child(0).label() + "</a></td>");
        } else if (item.child(indexDescription).numberOfChildren() > 0) {
            XMLTree description = item.child(indexDescription);
            out.print("<td>");
            if (indexLink >= 0
                    && item.child(indexLink).numberOfChildren() > 0) {
                XMLTree link = item.child(indexLink);
                out.print("<a href=\"" + link.child(0).label() + "\">");
            }
            out.println(description.child(0).label() + "</a></td>");
        } else {
            out.print("<td>No title available</td>");
        }

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

```

```java
    /**
     * Main method.
     *
     * @param args
     *            the command line arguments; unused here
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();

        /*
         * Input the source URL. https://news.yahoo.com/rss/.
         */
        out.print("Enter the URL of an RSS 2.0 news feed: ");
        String url = in.nextLine();
        XMLTree xml = new XMLTree1(url);

        /*
         * Asks user for the name of an output file including the .html
         * extension.
         */
        out.print(
                "Enter the the name of an output file including the \".html\" extension: ");
        String outFile = in.nextLine();
        SimpleWriter file = new SimpleWriter1L(outFile);

        String attribute = "";
        if (xml.label().equals("rss")) {
            attribute = xml.attributeValue("version");
        }

        XMLTree channel = xml.child(0);
        if (attribute.equals("2.0")) {
            outputHeader(channel, file);
            // item tag and its children
            for (int i = 0; i < channel.numberOfChildren(); i++) {
                if (channel.child(i).label().equals("item")) {
                    XMLTree item = channel.child(i);
                    processItem(item, file);
                }
            }

            outputFooter(file);
        }

        xml.display();

        in.close();
        out.close();
    }

}
```