```java
import components.simplereader.SimpleReader;

/**
 * CSE 2221 Project #4. Program to convert an XML RSS (version 2.0) feed from a
 * given URL into the corresponding HTML output file.
 *
 * @author Faye Leigh
 *
 */
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";

        /*
         * Get indices of title, link, and description tags
         */
        int titleIndex = getChildElement(channel, "title"),
                linkIndex = getChildElement(channel, "link"),
                descriptionIndex = getChildElement(channel, "description");

        /*
         * Get title, link, and description
         */
        String title, link, description;
        if (channel.child(titleIndex).numberOfChildren() > 0) {
            title = channel.child(titleIndex).child(0).label();
        } else {
            title = "No title";
        }
        if (channel.child(descriptionIndex).numberOfChildren() > 0) {
            description = channel.child(descriptionIndex).child(0).label();
        } else {
            description = "No description";
        }
        link = channel.child(linkIndex).child(0).label();

        /*
         * Start printing to HTML file
         */
        out.println("<html><head><title>" + title + "</title></head><body>");
        out.println("<h1>");
        out.println("<a href=" + link + ">" + title + "</a>");
        out.println("</h1>");
        out.println("<p>");
        out.println(description);
        out.println("</p>");
        out.println("<table border=\"1\">");
        out.println("<tr>");
        out.println("<th>Date</th>");
        out.println("<th>Source</th>");
        out.println("<th>News</th>");
        out.println("</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></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 index = -1, i = 0;
        boolean childFound = false;

        /*
         * Checks each child of xml and stops when matching tag label is found
         */
        while (i < xml.numberOfChildren() && !childFound) {
            if (xml.child(i).isTag() && xml.child(i).label() == tag) {
                index = i;
                childFound = true;
            }
            i++;
        }
        return index;
    }

     * Processes one news item and outputs one table row. The row contains three
    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 titleIndex = getChildElement(item, "title"),
                linkIndex = getChildElement(item, "link"),
                dateIndex = getChildElement(item, "pubDate"),
                sourceIndex = getChildElement(item, "source"),
                descriptionIndex = getChildElement(item, "description");

        out.println("<tr>"); //Start of table row

        /*
         * Checks that there is a publication date tag and that it is not empty.
         * Adds publication date to table if available
         */
        if (dateIndex > -1 && item.child(dateIndex).numberOfChildren() > 0) {
            out.println(
                    "<td>" + item.child(dateIndex).child(0).label() + "</td>");
        } else {
            out.println("<td>No date available</td>");
        }

        /*
         * Checks that there is a source tag and that it has a text child. If it
         * has both, adds the text to table with hyperlink to url. If only url
         * exists, adds url to table
         */
        if (sourceIndex > -1) {
            if (item.child(sourceIndex).numberOfChildren() > 0) {
                out.println("<td><a href=\""
                        + item.child(sourceIndex).attributeValue("url") + "\">"
                        + item.child(sourceIndex).child(0).label()
                        + "</a></td>");
            } else {
                out.println(
                        "<td>\"" + item.child(sourceIndex).attributeValue("url")
                                + "\"</td>");
            }
        } else {
```

```java
            out.println("<td>No source available</td>");
        }

        /*
         * Checks that there is a title and that it is not empty.
         */
        if (titleIndex > -1 && item.child(titleIndex).numberOfChildren() > 0) {

            /*
             * If link is available, adds title to table with hyperlink.
             * Otherwise, adds title to table with no hyperlink
             */
            if (linkIndex > -1
                    && item.child(linkIndex).numberOfChildren() > 0) {
                out.println("<td><a href=\""
                        + item.child(linkIndex).child(0).label() + "\">"
                        + item.child(titleIndex).child(0).label()
                        + "</a></td>");
            } else {
                out.println("<td>" + item.child(titleIndex).child(0).label()
                        + "</td>");
            }

            /*
             * If no title, checks that there is a description and that it is
             * not empty
             */
        } else if (descriptionIndex > -1
                && item.child(descriptionIndex).numberOfChildren() > 0) {
            /*
             * If link is available, adds description to table with hyperlink.
             * Otherwise, adds description to table with no hyperlink
             */
            if (linkIndex > -1
                    && item.child(linkIndex).numberOfChildren() > 0) {
                out.println("<td><a href=\""
                        + item.child(linkIndex).child(0).label() + "\">"
                        + item.child(descriptionIndex).child(0).label()
                        + "</a></td>");
            } else {
                out.println(
                        "<td>" + item.child(descriptionIndex).child(0).label()
                                + "</td>");
            }

            /*
             * If no title or description and link is available, adds hyperlink
             * to table. If none are available, no hyperlink is added
             */
        } else {
            if (linkIndex > -1
                    && item.child(linkIndex).numberOfChildren() > 0) {
                out.println("<td><a href=\""
                        + item.child(linkIndex).child(0).label()
                        + "\">No title available</a></td>");
            } else {
                out.println("<td>No title available</td>");
            }
        }
    }

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

        out.println("Please enter the URL of an RSS 2.0 feed");
        String url = in.nextLine();
        out.println("Please enter a name for the output file (no extension)");
        String outputFileName = in.nextLine() + ".html";

        XMLTree xml = new XMLTree1(url);
        SimpleWriter html = new SimpleWriter1L(outputFileName);
        int channelIndex = getChildElement(xml, "channel");

        outputHeader(xml.child(0), html);

        /*
         * Process each item tag in channel
         */
        for (int i = 0; i < xml.child(channelIndex).numberOfChildren(); i++) {
            if (xml.child(channelIndex).child(i).label() == "item") {
                processItem(xml.child(channelIndex).child(i), html);
            }
        }

        outputFooter(html);

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

}

```