```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 Michael Gorman
 *
 */
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";

        //initialize strings as desired values
        String title = "Empty title";
        String link = null;
        String description = "No description";

        //get indexes of each child element
        int titleInd = getChildElement(channel, "title");
        int linkInd = getChildElement(channel, "link");
        int descInd = getChildElement(channel, "description");

        /*
         * the following if statements check if title and description have
         * children and if so then they redefine the strings as the labels of
         * their respective children
         */
        if (channel.child(titleInd).numberOfChildren() > 0) {
            title = channel.child(titleInd).child(0).label();
        }

        if (channel.child(descInd).numberOfChildren() > 0) {
            description = channel.child(descInd).child(0).label();
        }

        //no if statement necessary because link is mandatory under channel
        link = channel.child(linkInd).child(0).label();

        //print statements to html file
        out.println("<html>");
        out.println("<head>");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1><a href=\"" + link + "\">" + title + "</a></h1>");
        out.println("<p>" + description + "</p>");
        out.println("<table border=\"1\">");

    }

    /**
     * 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";

        //closing statements
        out.println("</table>");
        out.print("</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";
        //initialize index
        int index = -1;
        //store total num of children for xml tree
        int childrenNum = xml.numberOfChildren();
        /*
         * while loop that will sift through all children of xml and if the
         * label of that child matches string tag then the index of that child
         * is stored in index
         */
        int i = 0;
        while (index == -1 && i < childrenNum) {
            if (xml.child(i).label().equals(tag)) {
                index = i;
            }
            i++;
        }
        //return index
        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";

        //initialize strings to default values
        String pubDate = "No date available";
        String source = "No source available";
        String titleDescription = "No title available";
        /*
         * source url and link must be initialized as empty which is important
         * later in if statements
         */
        String sourceURL = "";
        String link = "";

        //String description = "No description available";
        //get indexes which each element occurs
        int pubDateInd = getChildElement(item, "pubDate");
        int sourceInd = getChildElement(item, "source");
        int titleInd = getChildElement(item, "title");
        int urlInd = getChildElement(item, "link");
        int descInd = getChildElement(item, "description");
        //if source exists
        if (sourceInd != -1) {
            source = item.child(sourceInd).child(0).label();
            sourceURL = item.child(sourceInd).attributeValue("url");
        }
        /*
         * if else statement that checks if title has a value and if not checks
         * for a description and if not then it stays as no title available
         */
        if (titleInd != -1) {
            if (item.child(titleInd).numberOfChildren() > 0) {
                titleDescription = item.child(titleInd).child(0).label();
            }
        } else if (descInd != -1) {
            if (item.child(descInd).numberOfChildren() > 0) {
                titleDescription = item.child(descInd).child(0).label();
            }
        }

        if (pubDateInd != -1) {
            pubDate = item.child(pubDateInd).child(0).label();
        }

        if (urlInd != -1) {
            link = item.child(urlInd).child(0).label();
        }
        /*
         * if else if else statement that prints each item to a row and checks
         * for what links are provided and changes print statements accordingly
         */
        out.println("<tr>");
        if (!sourceURL.isEmpty() && !link.isEmpty()) {
            out.println("<td>" + pubDate + "</td>");
            out.println("<td>");
            out.println("<a href=\"" + sourceURL + "\">" + source + "</a>");
            out.println("</td>");
            out.println("<td>");
            out.println(
                    "<a href=\"" + link + "\">" + titleDescription + "</a>");
            out.println("</td>");
        } else if (!sourceURL.isEmpty()) {
            out.println("<td>" + pubDate + "</td>");
            out.println("<td>");
            out.println("<a href=\"" + sourceURL + "\">" + source + "</a>");
            out.println("</td>");
            out.println("<td>");
        } else {
            out.println("<td>" + pubDate + "</td>");
            out.println("<td>" + source + "</td>");
            out.println("<td>" + titleDescription + "</td>");
        }
        out.println("</tr>");
    }

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

        //prompt user for RSS URL
        out.print("Enter an RSS URL: ");
        String url = in.nextLine();

        //prompt user for html file name to write to
        out.print("Enter an html file name: ");
        String htmlFile = in.nextLine();
        //initialize new output stream that writes to file
        SimpleWriter fileOut = new SimpleWriter1L(htmlFile);

        //initialize xml tree off inputed url
        XMLTree rssWeb = new XMLTree1(url);

        /*
         * if the root is a tag and has version attribute of value 2.0 then
         * proceed to write to html file as desired output headers first and if
         * the tag label is "item" then process the item and add it to the html
         * file if RSS file is not in proper format print error statement
         */
        if (rssWeb.isTag() && rssWeb.hasAttribute("version")
                && rssWeb.attributeValue("version").equals("2.0")) {
            XMLTree channel = rssWeb.child(0);
            outputHeader(channel, fileOut);
            for (int i = 0; i < channel.numberOfChildren(); i++) {
                if (channel.child(i).label().equals("item")) {
                    processItem(channel.child(i), fileOut);
                }
            }
            outputFooter(fileOut);
        } else {
            out.println("Error with RSS file.");
        }

        //close input and output streams
        fileOut.close();
        in.close();
        out.close();
    }
}
```