```
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 Ansh Pachauri
 *
 */
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>
     * <ul>
     * The unordered list
     *
     *
     * @param feeds
     *            the channel element XMLTree
     * @param out
```

```java
     *            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 indexOutputHeader(XMLTree feeds, SimpleWriter out) {

        String title = feeds.attributeValue("title");

        out.println("<html>");
        out.println("<head>");
        out.println("<title>" + title + "</title>");
        out.println("<h1>" + title + "</h1>");
        out.println("</head>");
        out.println("<body>");
        out.println("<ul>");
    }

    /**
     * Outputs the "closing" tags in the generated HTML file. These are the
     * expected elements generated by this method:
     *
     * </ul>
     * </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 indexOutputFooter(SimpleWriter out) {

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

    /**
     * Processes one XML RSS (version 2.0) feed from a given URL converting it
     * into the corresponding HTML output file.
     *
     * @param url
     *            the URL of the RSS feed
     * @param file
     *            the name of the HTML output file
     * @param out
     *            the output stream to report progress or errors
     * @updates out.content
     * @requires out.is_open
     * @ensures <pre>
     * [reads RSS feed from url, saves HTML document with table of news items
     *   to file, appends to out.content any needed messages]
     * </pre>
     */
    private static void processFeed(String url, String file, SimpleWriter out) {
        XMLTree xml = new XMLTree1(url);
        SimpleWriter fileOutName = new SimpleWriter1L(file);

        outputHeader(xml.child(0), fileOutName);
        int i = 0;
        while (xml.child(0).numberOfChildren() > i) {
            if (xml.child(0).child(i).label().equals("item")) {
                processItem(xml.child(0).child(i), fileOutName);
            }
            i++;
        }
        outputFooter(fileOutName);
    }

    /**
     * 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) {
        /*
         * If title has child then the text will be assigned to String title,
         * otherwise, the String title will output "Empty Title".
         */

        String title;
        if (getChildElement(channel, "title") >= 0) {
            if (channel.child(getChildElement(channel, "title"))
                    .numberOfChildren() >= 1) {
                title = channel.child(getChildElement(channel, "title"))
                        .child(0).label();
            } else {
                title = "Empty Title";
            }
        } else if (getChildElement(channel, "description") >= 0) {
            if (channel.child(getChildElement(channel, "description"))
                    .numberOfChildren() >= 1) {
                title = channel.child(getChildElement(channel, "description"))
                        .label();
            } else {
                title = "Empty Title";
            }

        } else {
            title = "Empty Title";
        }

        /*
         * If description has child then the text will be assigned to String
         * desc, otherwise, the String desc will output "No description".
         */
        String desc = "";
        if (channel.child(getChildElement(channel, "description"))
                .numberOfChildren() >= 0) {
```

```
            if (channel.child(getChildElement(channel, "description"))
                    .numberOfChildren() >= 1) {
                desc = channel.child(getChildElement(channel, "description"))
                        .child(0).label();
            }

        } else {
            desc = "No Description.";
        }
        // header
        out.println("<html>");
        out.println("<head>");
        out.println("<title>" + title + "</title>");
        out.println("</head>");
        out.println("<body>");
        out.println(
                "<h1><a href=\""
                        + channel.child(getChildElement(channel, "link"))
                                .child(0).label()
                        + "\">" + title + "</a></h1>");
        out.println("<p>" + desc + "</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) {

        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) {
```

```java

        int index = -1;
        int i = 0;
        while (i < xml.numberOfChildren() && 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) {
        //table start
        out.println("<tr>");
        //assigns pubDate with the date then prints the row
```

```java
item
        String pubDate;
        if (getChildElement(item, "pubDate") >= 0) {
            if (item.child(getChildElement(item, "pubDate"))
                    .numberOfChildren() > 0) {
                pubDate = item.child(getChildElement(item, "pubDate")).child(0)
                        .label();
            } else {
                pubDate = "No date available";
            }

        } else {
            pubDate = "No date available";
        }
        out.println("<th>" + pubDate + "</th>");
        //assigns source with the source link then prints the row item
        String source;
        String sourceLink = "";
        int i = 0;
        if (getChildElement(item, "source") >= 0) {
            if (item.child(getChildElement(item, "source"))
                    .numberOfChildren() > 0) {
                source = item.child(getChildElement(item, "source")).child(0)
                        .label();
                if (item.child(getChildElement(item, "source"))
                        .hasAttribute("url")) {
                    sourceLink = item.child(getChildElement(item, "source"))
                            .attributeValue("url");
                    i = 1;
                } else {
                    i = 0;
                }

            } else {
                source = "No source available";
            }

        } else {
            source = "No source available";
        }

        if (i == 1) {
            out.println(
                    "<th><a href=\"" + sourceLink + "\">" + source + "</th>");
        } else {
            out.println("<th>" + source + "</th>");
        }
        //assigns title with the title of the article then prints the row item
        String title = "No title available";
        if (getChildElement(item, "title") >= 0) {
            if (item.child(getChildElement(item, "title"))
                    .numberOfChildren() > 0) {
                title = item.child(getChildElement(item, "title")).child(0)
                        .label();
            }

        } else if (getChildElement(item, "description") >= 0) {
            if (item.child(getChildElement(item, "description"))
                    .numberOfChildren() > 0) {
                title = item.child(getChildElement(item, "description"))
                        .child(0).label();
            }
        }

        String link = "";
        if (getChildElement(item, "link") >= 0) {
            if (item.child(getChildElement(item, "link"))
                    .numberOfChildren() >= 1) {
                link = item.child(getChildElement(item,
"link")).child(0)
                        .label();
            }
        }

        out.println("<th><a href=\"" + link + "\">" + title +
"</th>");

        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();

        out.print(
                "Enter the name of XML file containing URLs for
RSS v2.0 feeds: ");
        String url = in.nextLine();
        XMLTree xml = new XMLTree1(url);

        out.print("Enter output file name: ");
        String outFile = in.nextLine();

        SimpleWriter fileOut = new SimpleWriter1L(outFile);

        indexOutputHeader(xml, fileOut);

        int i = 0;
        while (xml.numberOfChildren() > i) {
            if (xml.child(i).label().equals("feed")) {
                processFeed(xml.child(i).attributeValue("url"),
            xml.child(i).attributeValue("file"), fileOut);
        fileOut.println("<li><a href=\""
            + xml.child(i).attributeValue("file") + "\">"
            + xml.child(i).attributeValue("name") + "</a></li>");

      }

      i++;
    }

    indexOutputFooter(fileOut);

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

}

```