```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;

/**
 * This program inputs an XML RSS (version 2.0) feed from a given URL and
 * outputs various elements of the feed to the console.
 *
 * @author Put your name here
 *
 */
public final class RSSProcessing {

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

    /**
     * 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 the {@code XMLTree} matching the
     *         given tag 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 the {@code XMLTree} matching the
     *   given tag 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";

        boolean found = false;
        XMLTree temp = xml;
        int i = 0;

        while (!found) {
            if (xml.child(i).label() == tag) {
                found = true;
            } else {
                i++;
            }
        }

        return i;

    }
```

```
60
61      /**
62       * Processes one news item and outputs the title, or the description if the
63       * title is not present, and the link (if available) with appropriate
64       * labels.
65       *
66       * @param item
67       *            the news item
68       * @param out
69       *            the output stream
70       * @requires [the label of the root of item is an <item> tag] and
71       *           out.is_open
72       * @ensures out.content = #out.content * [the title (or description) and
73       *          link]
74       */
75      private static void processItem(XMLTree item, SimpleWriter out) {
76          assert item != null : "Violation of: item is not null";
77          assert out != null : "Violation of: out is not null";
78          assert item.isTag() && item.label().equals("item") : ""
79                  + "Violation of: the label root of item is an <item> tag";
80          assert out.isOpen() : "Violation of: out.is_open";
81
82          System.out.println("Title: "
83                  + item.child(getChildElement(item, "title")).child(0));
84          System.out.println(
85                  "Link: " + item.child(getChildElement(item, "link")).child(0));
86
87      }
88
89      /**
90       * Main method.
91       *
92       * @param args
93       *            the command line arguments; unused here
94       */
95      public static void main(String[] args) {
96          /*
97           * Open I/O streams.
98           */
99          SimpleReader in = new SimpleReader1L();
100         SimpleWriter out = new SimpleWriter1L();
101         /*
102          * Input the source URL.
103          */
104         out.print("Enter the URL of an RSS 2.0 news feed: ");
105         String url = in.nextLine();
106         /*
107          * Read XML input and initialize XMLTree. If input is not legal XML,
108          * this statement will fail.
109          */
110         XMLTree xml = new XMLTree1(url);
111         /*
112          * Extract <channel> element.
113          */
114         XMLTree channel = xml.child(0);
115         /*
116          * Output title, link, and description
117          */
118         System.out.println("Title: "
```

```
119                 + channel.child(getChildElement(channel, "title")).child(0));
120         System.out.println("Description: " + channel
121                 .child(getChildElement(channel, "description")).child(0));
122         System.out.println("Link: "
123                 + channel.child(getChildElement(channel, "link")).child(0));
124 
125         for (int i = 0; i < channel.numberOfChildren(); i++) {
126             if (channel.child(i).label() == "item") {
127                 processItem(channel.child(i), out);
128             }
129         }
130 
131         /*
132          * Close I/O streams.
133          */
134         in.close();
135         out.close();
136     }
137 
138 }
```