```java
import components.simplereader.SimpleReader;

/**
 * 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";

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

        while (i > 0 && index < 0) {
            i--;
            if (xml.child(i).label().equals(tag)) {
                index = i;
            }
        }
        return index;
    }

    /**
     * Processes one news item and outputs the title, or the description if the
     * title is not present, and the link (if available) with appropriate
     * labels.
     *
     * @param item
     *            the news item
     * @param out
     *            the output stream
     * @requires [the label of the root of item is an <item> tag] and
     *           out.is_open
     * @ensures out.content = #out.content * [the title (or description) and
     *          link]
     */
    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";

        if (getChildElement(item, "title") >= 0) {
            XMLTree title = item.child(getChildElement(item, "title"));
            if (title.numberOfChildren() > 0) {
                XMLTree titleChild0 = title.child(0);
                out.println("Title: " + titleChild0);
            }
        } else if (getChildElement(item, "description") >= 0) {
            XMLTree description = item
                    .child(getChildElement(item, "description"));
            if (description.numberOfChildren() > 0) {
                XMLTree descriptionVal = description.child(0);
                out.println("Description: " + descriptionVal);
            }
        }

        if (getChildElement(item, "link") >= 0) {
            XMLTree link = item.child(getChildElement(item, "link"));
            XMLTree linkVal = link.child(0);
            out.println("Link: " + linkVal);
        }
    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments; unused here
     */
    public static void main(String[] args) {
        /*
         * Open I/O streams.
         */
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();
        /*
         * Input the source URL.
         */
        out.print("Enter the URL of an RSS 2.0 news feed: ");
        String url = in.nextLine();
        /*
         * Read XML input and initialize XMLTree. If input is not legal XML,
         * this statement will fail.
         */
        XMLTree xml = new XMLTree1(url);
        /*
         * Extract <channel> element.
         */
        XMLTree channel = xml.child(0);
        XMLTree title;
        XMLTree description;
        XMLTree link;

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

        if (title.numberOfChildren() > 0) {
            out.println("Title: " + title.child(0).label());
        } else {
            out.println("Title is blank.");
        }

        //Description
        int descriptionNum = getChildElement(channel, "description");

        description = channel.child(descriptionNum);
        if (description.numberOfChildren() > 0) {
            out.println("Description: " + description.child(0).label());
        } else {
            out.println("Description is blank.");
        }

        //Link
        int linkNum = getChildElement(channel, "link");

        link = channel.child(linkNum);
        out.println("Link: " + link.child(0).label());

        /*
         * TODO: #4 - for each item, output title (or description, if title is
         * not available) and link (if available)
         */
        int itemNum = getChildElement(channel, "item");
        if (itemNum >= 0) {
            XMLTree item = channel.child(itemNum);
            processItem(item, out);
        }

        /*
         * Close I/O streams.
         */
        in.close();
        out.close();
    }

}
```