```java
import components.simplereader.SimpleReader;
import components.simplereader.SimpleReader1L;
import components.simplewriter.SimpleWriter;
import components.simplewriter.SimpleWriter1L;
import components.xmltree.XMLTree;
import components.xmltree.XMLTree2;

/**
 * Program to practice recursion on {@code XMLTree}s.
 *
 * @author Vaishnavi Kasabwala
 *
 */
public final class XMLTreeRecursion {

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

    /**
     * Returns the number of occurrences of the given tag in the given
     * {@code XMLTree}.
     *
     * @param xml
     *            the {@code XMLTree}
     * @param tag
     *            the tag name
     * @return the number of occurrences of the given tag in the given
     *         {@code XMLTree}
     * @ensures <pre>
     * tagCount =
     *    [the number of occurrences of the given tag in the given {@code XMLTree}]
     * </pre>
     */
    private static int tagCount(XMLTree xml, String tag) {
        assert xml != null : "Violation of: xml is not null";
        assert tag != null : "Violation of: tag is not null";

        int count = 0;
        if (xml.isTag()) {
            if (xml.label().equals(tag)) {
                count++;
            }
            for (int i = 0; i < xml.numberOfChildren(); i++) {
                count += tagCount(xml.child(i), tag);
            }
        }
        return count;
    }

    /**
     * Outputs the text nodes in the given {@code XMLTree} on separate lines.
     *
     * @param xml
     *            the {@code XMLTree}
```

```
58      * @param out
59      *            the output stream
60      * @updates out.content
61      * @requires out.is_open
62      * @ensures <pre>
63      * out.content = #out.content * [the text nodes of xml on separate lines]
64      * </pre>
65      */
66     private static void outputTextNodes(XMLTree xml, SimpleWriter out) {
67         assert xml != null : "Violation of: xml is not null";
68         assert out != null : "Violation of: out is not null";
69         assert out.isOpen() : "Violation of: out.is_open";
70
71         if (!xml.isTag()) {
72             out.println(xml.label());
73         } else {
74             for (int i = 0; i < xml.numberOfChildren(); i++) {
75                 outputTextNodes(xml.child(i), out);
76             }
77         }
78     }
79
80     /**
81      * Outputs n spaces.
82      *
83      * @param n
84      *            the number of spaces
85      * @param out
86      *            the output stream
87      * @updates out.content
88      * @requires out.is_open and n >= 0
89      * @ensures out.content = #out.content * [n spaces]
90      */
91     private static void outputSpaces(int n, SimpleWriter out) {
92         assert out != null : "Violation of: out is not null";
93         assert out.isOpen() : "Violation of: out.is_open";
94         assert n >= 0 : "Violation of: n >= 0";
95
96         // TODO - fill in body
97
98     }
99
100    /**
101     * Outputs the attributes ( name="value") of the given {@code XMLTree}'s
102     * root node to the given output stream.
103     *
104     * @param xml
105     *            the {@code XMLTree}
106     * @param out
107     *            the output stream
108     * @updates out.content
109     * @requires out.is_open and [the label of the root of xml is a tag]
110     * @ensures <pre>
111     * out.content =
112     *    #out.content * [the attributes ( name="value") of the root of xml]
113     * </pre>
114     */
```

```
115     private static void outputAttributes(XMLTree xml, SimpleWriter out) {
116         assert xml != null : "Violation of: xml is not null";
117         assert out != null : "Violation of: out is not null";
118         assert xml
119                 .isTag() : "Violation of: the label of the root of xml is a tag";
120         assert out.isOpen() : "Violation of: out.is_open";
121
122         // TODO - fill in body
123
124     }
125
126     /**
127      * Output the XML textual representation of the given {@code XMLTree}.
128      *
129      * @param xml
130      *            the {@code XMLTree}
131      * @param out
132      *            the output stream
133      * @param indentationLevel
134      *            the level of indentation
135      * @updates out.content
136      * @requires out.is_open and indentationLevel >= 0
137      * @ensures <pre>
138      * out.content = #out.content * [the XML textual representation of xml]
139      * </pre>
140      */
141     private static void outputXML(XMLTree xml, SimpleWriter out,
142             int indentationLevel) {
143         assert xml != null : "Violation of: xml is not null";
144         assert out != null : "Violation of: out is not null";
145         assert out.isOpen() : "Violation of: out.is_open";
146         assert indentationLevel >= 0 : "Violation of: indentationLevel >= 0";
147
148         // TODO - fill in body
149
150     }
151
152     /**
153      * Main method.
154      *
155      * @param args
156      *            the command line arguments
157      */
158     public static void main(String[] args) {
159         SimpleReader in = new SimpleReader1L();
160         SimpleWriter out = new SimpleWriter1L();
161
162         out.print("Enter a URL or file name for an XML source: ");
163         String url = in.nextLine();
164         XMLTree xml = new XMLTree2(url);
165
166         out.print("Enter the name of a tag: ");
167         String tag = in.nextLine();
168         while (!tag.equals("")) {
169             int count = tagCount(xml, tag);
170             out.println("The tag <" + tag + "> appears " + count + " times.");
171             out.println();
```

```
172             out.print("Enter the name of a tag: ");
173             tag = in.nextLine();
174         }
175 
176         out.println();
177         out.println("The text nodes:");
178         outputTextNodes(xml, out);
179 
180         //out.println();
181         //out.println("The XML:");
182         //outputXML(xml, out, 0);
183 
184         //https://news.yahoo.com/rss/.
185 
186         in.close();
187         out.close();
188     }
189 
190 }
```