```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- excerpted from http://www.w3schools.com/xquery/xquery_example.asp -->
<bib>
        <book year="1994">
                <title>TCP/IP Illustrated</title>
                <author>
                        <last>Stevens</last>
                        <first>W.</first>
                </author>
                <publisher>Addison-Wesley</publisher>
                <price>65.95</price>
        </book>

        <book year="1992">
                <title>Advanced Programming in the Unix environment</title>
                <author>
                        <last>Stevens</last>
                        <first>W.</first>
                </author>
                <publisher>Addison-Wesley</publisher>
                <price>65.95</price>
        </book>

        <book year="2000">
                <title>Data on the Web</title>
                <author>
                        <last>Abiteboul</last>
                        <first>Serge</first>
                </author>
                <author>
                        <last>Buneman</last>
                        <first>Peter</first>
                </author>
                <author>
                        <last>Suciu</last>
                        <first>Dan</first>
                </author>
                <publisher>Morgan Kaufmann Publishers</publisher>
                <price>39.95</price>
        </book>

        <book year="1999">
                <title>The Technology and Content for Digital TV</title>
                <editor>
                        <last>Gerbarg</last>
                        <first>Darcy</first>
                        <affiliation>CITI</affiliation>
                </editor>
                <publisher>Kluwer Academic Publishers</publisher>
                <price>129.95</price>
        </book>
</bib>

```

```
 1: <!ELEMENT  emails  (message+)>
 2: <!ELEMENT  message  (subject?, body, reply*)>
 3:   <!ATTLIST  message
 4:     num   ID     #REQUIRED
 5:     to    CDATA  #REQUIRED
 6:     from  CDATA  #FIXED     "brenda&#64;xyzcompany.com"
 7:     date  CDATA  #REQUIRED>
 8: <!ELEMENT  subject  EMPTY>
 9:   <!ATTLIST   subject
10:     title  CDATA  #IMPLIED>
11: <!ELEMENT  body   ANY>
12: <!ELEMENT  reply  EMPTY>
13:   <!ATTLIST  reply
14:     status  (yes | no)  "no">
```

```
 1: <?xml version="1.0" standalone="no"?>
 2: <!-- from http://wps.aw.com/aw_webwizard/0,6065,184738-,00.html -->
 3: <!DOCTYPE  emails SYSTEM "emails.dtd">
 4: <emails>
 5:   <message num="a1"
 6:     to="joe&#64;acmeshipping.com"
 7:     from="brenda&#64;xyzcompany.com"
 8:     date="02/09/01">
 9:   <subject title="Order 10011" />
10:   <body>
11:     Joe,
12:       Please let me know if order number 10011 has shipped.
13:     Thanks,
14:     Brenda
15:   </body>
16:   <reply status="yes"/>
17:   </message>
18: </emails>
```

```
 1: <?xml version="1.0" standalone="yes"?>
 2:
 3: <!DOCTYPE foo [
 4:   <!ELEMENT foo (bar,baz)>
 5:   <!ELEMENT bar (#PCDATA)>
 6:   <!ELEMENT baz (#PCDATA)>
 7: ]>
 8:
 9: <!-- notice that foo contains ignorable whitespace -->
10:
11: <foo>
12:   <bar/>
13:   <baz/>
14: </foo>
```

```
 1: <?xml version="1.0" standalone="yes"?>
 2:
 3: <!DOCTYPE foo [
 4:   <!ELEMENT foo (bar,baz)>
 5:   <!ELEMENT bar EMPTY>
 6:   <!ELEMENT baz EMPTY>
 7: ]>
 8:
 9: <!-- notice that the ordering of baz and bar is invalid! -->
10:
11: <foo>
12:   <baz/>
13:   <bar/>
14: </foo>
```

```
 1: import javax.xml.parsers.SAXParser;
 2: import javax.xml.parsers.SAXParserFactory;
 3: import org.xml.sax.helpers.DefaultHandler;
 4: import org.xml.sax.SAXParseException;
 5:
 6:
 7: /**
 8:  * Lecture 8's demonstration of validation.
 9:  *
10:  * @author  Computer Science E-259
11:  **/
12:
13: public class SAXValidator extends DefaultHandler
14: {
15:     /**
16:      * Main driver.  Expects one command-line argument: the name of the
17:      * XML file to validate
18:      *
19:      * @param argv [0] - filename
20:      */
21:
22:     public static void main(String [] argv)
23:     {
24:         // grab filename
25:         String input = argv[0];
26:
27:         try
28:         {
29:             // instantiate a SAX parser factory
30:             SAXParserFactory factory = SAXParserFactory.newInstance();
31:
32:             // enable validation
33:             factory.setValidating(true);
34:
35:             // instantiate a SAX parser
36:             SAXParser parser = factory.newSAXParser();
37:
38:             // instantiate our little handler
39:             SAXValidator handler = new SAXValidator();
40:
41:             // parse the file
42:             parser.parse(input, handler);
43:         }
44:         catch (Exception e)
45:         {
46:             e.printStackTrace();
47:         }
48:     }
49:
50:
51:     /**
52:      * Receive notification of a recoverable parser error.
53:      *
54:      * @param e  the exception thrown
55:      */
56:
57:     public void error(SAXParseException e)
58:     {
59:         System.out.println("Parsing error at " + e.getLineNumber() +
60:                            ":" + e.getColumnNumber() +
61:                            ": " + e.getMessage());
62:     }
63:
64:
```

```
65:     /**
66:      * Receive notification of a parser warning.
67:      *
68:      * @param e  the exception thrown
69:      */
70:
71:     public void warning(SAXParseException e)
72:     {
73:         System.out.println("Parsing warning at " + e.getLineNumber() +
74:                            ":" + e.getColumnNumber() +
75:                            ": " + e.getMessage());
76:     }
77:
78:
79:     /**
80:      * Report a fatal XML parsing error.
81:      *
82:      * @param e  the exception thrown
83:      */
84:
85:     public void fatalError(SAXParseException e)
86:     {
87:         System.out.println("Fatal parsing warning at " + e.getLineNumber() +
88:                            ":" + e.getColumnNumber() +
89:                            ": " + e.getMessage());
90:     }
91: }
```

```
 1: <?xml version="1.0" standalone="yes"?>
 2:
 3: <!DOCTYPE foo [
 4:   <!ELEMENT foo ANY>
 5:   <!ELEMENT bar EMPTY>
 6:   <!ELEMENT baz EMPTY>
 7: ]>
 8:
 9: <!-- notice that foo contains significant whitespace! -->
10:
11: <foo>
12:   <bar/>
13:   <baz/>
14: </foo>
```

```
    1: <?xml version="1.0" standalone="yes"?>
    2:
    3: <!DOCTYPE SONG [
    4:  <!ELEMENT SONG (TITLE, COMPOSER+, PRODUCER*, PUBLISHER*, LENGTH?, YEAR?, ARTIST
+)>
    5:  <!ELEMENT TITLE (#PCDATA)>
    6:  <!ELEMENT COMPOSER (#PCDATA)>
    7:  <!ELEMENT PRODUCER (#PCDATA)>
    8:  <!ELEMENT PUBLISHER (#PCDATA)>
    9:  <!ELEMENT LENGTH (#PCDATA)>
   10:  <!ELEMENT YEAR (#PCDATA)>
   11:  <!ELEMENT ARTIST (#PCDATA)>
   12: ]>
   13:
   14: <SONG>
   15:   <TITLE>Everyday</TITLE>
   16:   <COMPOSER>Dave</COMPOSER>
   17:   <COMPOSER>Boyd Tinsley</COMPOSER>
   18:   <PRODUCER>Dave Matthews</PRODUCER>
   19:   <PUBLISHER>BMG</PUBLISHER>
   20:   <LENGTH>12:20</LENGTH>
   21:   <YEAR>2001</YEAR>
   22:   <ARTIST>Dave Matthews Band</ARTIST>
   23: </SONG>
```

```
 1: <?xml version="1.0" standalone="no"?>
 2:
 3: <!DOCTYPE SONG SYSTEM "song.dtd">
 4:
 5: <SONG>
 6:   <TITLE>Everyday</TITLE>
 7:   <COMPOSER>Dave</COMPOSER>
 8:   <COMPOSER>Boyd Tinsley</COMPOSER>
 9:   <PRODUCER>Dave Matthews</PRODUCER>
10:   <PUBLISHER>BMG</PUBLISHER>
11:   <LENGTH>12:20</LENGTH>
12:   <YEAR>2001</YEAR>
13:   <ARTIST>Dave Matthews Band</ARTIST>
14: </SONG>
```

```
    1: <!ELEMENT SONG (TITLE, COMPOSER+, PRODUCER*, PUBLISHER*, LENGTH?, YEAR?, ARTIST+
)>
    2: <!ELEMENT TITLE (#PCDATA)>
    3: <!ELEMENT COMPOSER (#PCDATA)>
    4: <!ELEMENT PRODUCER (#PCDATA)>
    5: <!ELEMENT PUBLISHER (#PCDATA)>
    6: <!ELEMENT LENGTH (#PCDATA)>
    7: <!ELEMENT YEAR (#PCDATA)>
    8: <!ELEMENT ARTIST (#PCDATA)>
```

```
 1: import javax.xml.parsers.SAXParser;
 2: import javax.xml.parsers.SAXParserFactory;
 3: import org.xml.sax.Attributes;
 4: import org.xml.sax.ContentHandler;
 5: import org.xml.sax.SAXException;
 6: import org.xml.sax.helpers.AttributesImpl;
 7: import org.xml.sax.helpers.DefaultHandler;
 8: import org.xml.sax.SAXParseException;
 9:
10:
11: /**
12:  * Lecture 8's demonstration of whitespace handling.
13:  *
14:  * @author  Computer Science E-259
15:  **/
16:
17: public class WhitespaceDemo extends DefaultHandler
18: {
19:     /**
20:      * Main driver.  Expects one command-line argument:
21:      * the name of the file to parse.
22:      *
23:      * @param argv [0] - filename
24:      */
25:
26:     public static void main(String [] argv)
27:     {
28:         // grab filename
29:         String input = argv[0];
30:
31:         try
32:         {
33:             // instantiate a SAX parser factory
34:             SAXParserFactory factory = SAXParserFactory.newInstance();
35:
36:             // enable validation
37:             factory.setValidating(true);
38:
39:             // instantiate a SAX parser
40:             SAXParser parser = factory.newSAXParser();
41:
42:             // instantiate our little handler
43:             WhitespaceDemo handler = new WhitespaceDemo();
44:
45:             // parse the file
46:             parser.parse(input, handler);
47:         }
48:         catch (Exception e)
49:         {
50:             e.printStackTrace();
51:         }
52:     }
53:
54:
55:     /**
56:      * Report a startElement event.
57:      *
58:      * @param   uri namespace
59:      * @param   localName   name of element, sans namespace
60:      * @param   qName       name of element, with namespace
61:      * @param   attributes  element's collection of attributes
62:      *
63:      * @throws  SAXException    general SAX error or warning
64:      */
```

```
65:
66:     public void startElement(String uri, String localName,
67:                              String qName, Attributes atts)
68:      throws SAXException
69:     {
70:         System.out.print("startElement(\"" + qName + ", {");
71:         for (int i = 0; i < atts.getLength(); i++)
72:         {
73:             System.out.print("(\"" + atts.getQName(i) + "\", \"" +
74:                              atts.getValue(i) + "\")");
75:             if (i != atts.getLength() - 1)
76:                 System.out.print(", ");
77:         }
78:         System.out.println("});");
79:     }
80:
81:
82:     /**
83:      * Report a characters event.
84:      *
85:      * @param   ch      characters
86:      * @param   start   start position in the character array
87:      * @param   length  number of characters to use from the character array
88:      *
89:      * @throws  SAXException    general SAX error or warning
90:      */
91:
92:     public void characters(char[] ch, int start, int length)
93:      throws SAXException
94:     {
95:         System.out.println("characters(\"" + new String(ch, start, length) +
96:                            "\");");
97:     }
98:
99:
100:    /**
101:     * Report an endElement event.
102:     *
103:     * @param   uri         namespace
104:     * @param   localName   name of element, sans namespace
105:     * @param   qName       name of element, with namespace
106:     *
107:     * @throws  SAXException    general SAX error or warning
108:     */
109:
110:    public void endElement(String uri, String localName, String qName)
111:     throws SAXException
112:    {
113:        System.out.println("endElement(\"" + qName + "\");");
114:    }
115:
116:
117:    /**
118:     * Report a startDocument event.
119:     */
120:
121:    public void startDocument() throws SAXException
122:    {
123:        System.out.println("\nstartDocument();");
124:    }
125:
126:
127:    /**
128:     * Report an endDocument event.
```

```
129:      *
130:      * @throws  SAXException    general SAX error or warning
131:      */
132:
133:     public void endDocument() throws SAXException
134:     {
135:         System.out.println("endDocument();\n");
136:     }
137:
138:
139:     /**
140:      * Receive notification of a recoverable parser error.
141:      *
142:      * @param e  the exception thrown
143:      */
144:
145:     public void error (SAXParseException e)
146:     {
147:         System.out.println("Parsing error:  " + e.getMessage());
148:     }
149:
150:
151:     /**
152:      * Receive notification of a parser warning.
153:      *
154:      * @param e  the exception thrown
155:      */
156:
157:     public void warning (SAXParseException e)

158:     {
159:         System.out.println("Parsing warning:  " + e.getMessage());
160:     }
161:
162:
163:     /**
164:      * Report a fatal XML parsing error.
165:      *
166:      * @param e  the exception thrown
167:      */
168:
169:     public void fatalError (SAXParseException e)
170:     {
171:         System.out.println("Fatal parsing error:  " + e.getMessage());
172:         System.exit(1);
173:     }
174: }
```

```
    1: <?xml version="1.0" encoding="iso-8859-1"?>
    2:
    3: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.or
g/TR/xhtml1/DTD/xhtml1-transitional.dtd">
    4:
    5: <html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
    6:     <head>
    7:         <title/>
    8:     </head>
    9:     <body/>
   10: </html>
```