

```
1: <?xml version="1.0" encoding="ISO-8859-1"?>
2: <!-- excerpted from http://www.w3schools.com/xquery/xquery_example.asp -->
3: <bib>
4:     <book year="1994">
5:         <title>TCP/IP Illustrated</title>
6:         <author>
7:             <last>Stevens</last>
8:             <first>W.</first>
9:         </author>
10:        <publisher>Addison-Wesley</publisher>
11:        <price>65.95</price>
12:    </book>
13:
14:    <book year="1992">
15:        <title>Advanced Programming in the Unix environment</title>
16:        <author>
17:            <last>Stevens</last>
18:            <first>W.</first>
19:        </author>
20:        <publisher>Addison-Wesley</publisher>
21:        <price>65.95</price>
22:    </book>
23:
24:    <book year="2000">
25:        <title>Data on the Web</title>
26:        <author>
27:            <last>Abiteboul</last>
28:            <first>Serge</first>
29:        </author>
30:        <author>
31:            <last>Buneman</last>
32:            <first>Peter</first>
33:        </author>
34:        <author>
35:            <last>Suciu</last>
36:            <first>Dan</first>
37:        </author>
38:        <publisher>Morgan Kaufmann Publishers</publisher>
39:        <price>39.95</price>
40:    </book>
41:
42:    <book year="1999">
43:        <title>The Technology and Content for Digital TV</title>
44:        <editor>
45:            <last>Gerbarg</last>
```

```
46:                <first>Darcy</first>
47:                <affiliation>CITI</affiliation>
48:            </editor>
49:            <publisher>Kluwer Academic Publishers</publisher>
50:            <price>129.95</price>
51:        </book>
52: </bib>
53:
```

```
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 XML file to
17:      * 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:         }
45:         catch (Exception e)
```

```
46:         {
47:             e.printStackTrace();
48:         }
49:     }
50:
51:
52:     /**
53:      * Receive notification of a recoverable parser error.
54:      *
55:      * @param e the exception thrown
56:      */
57:
58:     public void error(SAXParseException e)
59:     {
60:         System.out.println("Parsing error at " + e.getLineNumber() +
61:                             ":" + e.getColumnNumber() +
62:                             ": " + e.getMessage());
63:     }
64:
65:
66:     /**
67:      * Receive notification of a parser warning.
68:      *
69:      * @param e the exception thrown
70:      */
71:
72:     public void warning(SAXParseException e)
73:     {
74:         System.out.println("Parsing warning at " + e.getLineNumber() +
75:                             ":" + e.getColumnNumber() +
76:                             ": " + e.getMessage());
77:     }
78:
79:
80:     /**
81:      * Report a fatal XML parsing error.
82:      *
83:      * @param e the exception thrown
84:      */
85:
86:     public void fatalError(SAXParseException e)
87:     {
88:         System.out.println("Fatal parsing warning at " + e.getLineNumber() +
89:                             ":" + e.getColumnNumber() +
90:                             ": " + e.getMessage());
```



```
91:     }  
92: }
```

```
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:     }
49:     catch (Exception e)
50:     {
51:         e.printStackTrace();
52:     }
53: }
54:
55:
56: /**
57:  * Report a startElement event.
58:  *
59:  * @param uri namespace
60:  * @param localName name of element, sans namespace
61:  * @param qName name of element, with namespace
62:  * @param attributes element's collection of attributes
63:  *
64:  * @throws SAXException general SAX error or warning
65:  */
66:
67: public void startElement(String uri, String localName,
68:                         String qName, Attributes atts)
69:     throws SAXException
70: {
71:     System.out.print("startElement(\"" + qName + ", {");
72:     for (int i = 0; i < atts.getLength(); i++)
73:     {
74:         System.out.print("(" + atts.getQName(i) + "\", \"" +
75:                         atts.getValue(i) + "\"");
76:         if (i != atts.getLength() - 1)
77:             System.out.print(", ");
78:     }
79:     System.out.println("}");
80: }
81:
82:
83: /**
84:  * Report a characters event.
85:  *
86:  * @param ch characters
87:  * @param start start position in the character array
88:  * @param length number of characters to use from the character array
89:  *
90:  * @throws SAXException general SAX error or warning
```

```
91:      */
92:
93:      public void characters(char[] ch, int start, int length)
94:          throws SAXException
95:      {
96:          System.out.println("characters(\"" + new String(ch, start, length) +
97:                             "\"");
98:      }
99:
100:
101:      /**
102:       * Report an endElement event.
103:       *
104:       * @param uri      namespace
105:       * @param localName name of element, sans namespace
106:       * @param qName     name of element, with namespace
107:       *
108:       * @throws SAXException general SAX error or warning
109:       */
110:
111:      public void endElement(String uri, String localName, String qName)
112:          throws SAXException
113:      {
114:          System.out.println("endElement(\"" + qName + "\"");
115:      }
116:
117:
118:      /**
119:       * Report a startDocument event.
120:       */
121:
122:      public void startDocument() throws SAXException
123:      {
124:          System.out.println("\nstartDocument()");
125:      }
126:
127:
128:      /**
129:       * Report an endDocument event.
130:       *
131:       * @throws SAXException general SAX error or warning
132:       */
133:
134:      public void endDocument() throws SAXException
135:      {
```



```
136:         System.out.println("endDocument();\n");
137:     }
138:
139:
140:     /**
141:      * Receive notification of a recoverable parser error.
142:      *
143:      * @param e the exception thrown
144:      */
145:
146:     public void error (SAXParseException e)
147:     {
148:         System.out.println("Parsing error:  " + e.getMessage());
149:     }
150:
151:
152:     /**
153:      * Receive notification of a parser warning.
154:      *
155:      * @param e the exception thrown
156:      */
157:
158:     public void warning (SAXParseException e)
159:     {
160:         System.out.println("Parsing warning:  " + e.getMessage());
161:     }
162:
163:
164:     /**
165:      * Report a fatal XML parsing error.
166:      *
167:      * @param e the exception thrown
168:      */
169:
170:     public void fatalError (SAXParseException e)
171:     {
172:         System.out.println("Fatal parsing error:  " + e.getMessage());
173:         System.exit(1);
174:     }
175: }
```

```
1: <?xml version="1.0" encoding="iso-8859-1"?>
2:
3: <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/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>
```