```
1: <?xml version="1.0"?>
2: <foo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3:      xsi:noNamespaceSchemaLocation="foo.xsd">
4:         this
5:         is a
6:         test
7: </foo>
```

```
 1: <?xml  version="1.0"?>
 2:
 3: <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 4:
 5:   <xsd:element name="foo">
 6:     <xsd:simpleType>
 7:       <xsd:restriction base="xsd:string">
 8:         <xsd:whiteSpace value="preserve"/>
 9:       </xsd:restriction>
10:     </xsd:simpleType>
11:   </xsd:element>
12:
13: </xsd:schema>
```

```
 1: import javax.xml.parsers.SAXParser;
 2: import javax.xml.parsers.SAXParserFactory;
 3:
 4: import org.xml.sax.helpers.DefaultHandler;
 5:
 6: import org.xml.sax.Attributes;
 7: import org.xml.sax.SAXException;
 8: import org.xml.sax.SAXParseException;
 9:
10:
11: /**
12:  * Lecture 10's demonstration of validation
13:  * by XML DTD or XML Schema.  Just like Lecture 9's,
14:  * except that SAX events are reported, whitespace
15:  * characters are displayed as "\n", "\t", and "\r"
16:  * explicitly, and line numbers are reported in errors.
17:  *
18:  * @author  Computer Science E-259
19:  **/
20:
21: public class SAXValidator3 extends DefaultHandler
22: {
23:     /**
24:      * Main driver.  Expects one command-line argument: the name of the
25:      * XML file to validate
26:      *
27:      * @param argv [0] - filename
28:      */
29:
30:     public static void main(String [] argv)
31:     {
32:         if (argv.length == 0)
33:         {
34:             System.out.println("Usage: SAXValidator3 file [dtd|xsd]");
35:             System.exit(1);
36:         }
37:
38:         // grab filename
39:         String input = argv[0];
40:
41:         // grab validation mechanism, if any
42:         String validator = (argv.length > 1) ? argv[1] : null;
43:
44:         try
45:         {
46:             // instantiate a reference to a SAX parser
47:             SAXParser parser = null;
48:
49:             // instantiate a SAX parser factory
50:             SAXParserFactory factory = SAXParserFactory.newInstance();
51:
52:             // instantiate a SAX parser, enabling validation as requested
53:             if (validator != null && validator.equals("dtd"))
54:             {
55:                 factory.setValidating(true);
56:                 parser = factory.newSAXParser();
57:                 System.out.println("Validation by DTD on.");
58:             }
59:             else if (validator != null && validator.equals("xsd"))
60:             {
61:                 factory.setNamespaceAware(true);
62:                 factory.setValidating(true);
63:                 parser = factory.newSAXParser();
64:                 parser.setProperty
65:                  (
66:                   "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
67:                   "http://www.w3.org/2001/XMLSchema"
68:                  );
69:                 System.out.println("Validation by XML Schema on.");
70:             }
71:             else
72:             {
73:                 factory.setValidating(false);
74:                 parser = factory.newSAXParser();
75:                 System.out.println("Validation off.");
76:             }
77:
78:             // instantiate our little handler
79:             SAXValidator3 handler = new SAXValidator3();
80:
81:             // parse the file
82:             parser.parse(input, handler);
83:
84:         } catch (Exception e) {
85:             e.printStackTrace();
86:         }
87:     }
88:
89:     /**
90:      * Report a startElement event.
91:      *
92:      * @param   uri namespace
93:      * @param   localName   name of element, sans namespace
94:      * @param   qName       name of element, with namespace
95:      * @param   attributes  element's collection of attributes
96:      *
97:      * @throws  SAXException    general SAX error or warning
98:      */
99:
100:    public void startElement(String uri, String localName,
101:                             String qName, Attributes atts)
102:     throws SAXException
103:    {
104:        System.out.print("startElement(\"" + qName + ", {");
105:        for (int i = 0; i < atts.getLength(); i++)
106:        {
107:            System.out.print("(\"" + atts.getQName(i) + "\", \"" +
108:                             atts.getValue(i) + "\")");
109:            if (i != atts.getLength() - 1)
110:                System.out.print(", ");
111:        }
112:        System.out.println("});");
113:    }
114:
115:
116:    /**
117:     * Report a characters event.
118:     *
119:     * @param   ch      characters
120:     * @param   start   start position in the character array
121:     * @param   length  number of characters to use from the character array
122:     *
123:     * @throws  SAXException    general SAX error or warning
124:     */
```

```
125:
126:     public void characters(char[] ch, int start, int length)
127:      throws SAXException
128:     {
129:         String s = new String(ch, start, length);
130:         s = s.replaceAll("\n", "\\\\n");
131:         s = s.replaceAll("\t", "\\\\t");
132:         s = s.replaceAll("\r", "\\\\r");
133:         System.out.println("characters(\"" + s + "\");");
134:     }
135:
136:
137:     /**
138:      * Report an endElement event.
139:      *
140:      * @param   uri         namespace
141:      * @param   localName   name of element, sans namespace
142:      * @param   qName       name of element, with namespace
143:      *
144:      * @throws  SAXException    general SAX error or warning
145:      */
146:
147:     public void endElement(String uri, String localName, String qName)
148:      throws SAXException
149:     {
150:         System.out.println("endElement(\"" + qName + "\");");
151:     }
152:
153:
154:     /**
155:      * Report a startDocument event.
156:      */
157:
158:     public void startDocument() throws SAXException
159:     {
160:         System.out.println("\nstartDocument();");
161:     }
162:
163:
164:     /**
165:      * Report an endDocument event.
166:      *
167:      * @throws  SAXException    general SAX error or warning
168:      */
169:
170:     public void endDocument() throws SAXException
171:     {
172:         System.out.println("endDocument();\n");
173:     }
174:
175:
176:     /**
177:      * Receive notification of a parser warning.
178:      *
179:      * @param e  the exception thrown
180:      */
181:
182:     public void warning (SAXParseException e)
183:     {
184:         System.out.println("Parsing warning at line " + e.getLineNumber() +
185:                            ":  " + e.getMessage());
186:     }
187:
188:
189:     /**
190:      * Receive notification of a recoverable parser error.
191:      *
192:      * @param e  the exception thrown
193:      */
194:
195:     public void error (SAXParseException e)
196:     {
197:         System.out.println("Parsing error at line " + e.getLineNumber() +
198:                            ": " + e.getMessage());
199:     }
200:
201:
202:     /**
203:      * Report a fatal XML parsing error.
204:      *
205:      * @param e  the exception thrown
206:      */
207:
208:     public void fatalError (SAXParseException e)
209:     {
210:         System.out.println("Fatal parsing error at line " + e.getLineNumber()
+
211:                            ":  " + e.getMessage());
212:         System.exit(1);
213:     }
214: }
```