```
 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:
 9:
10: /**
11:  * Lecture 2's demonstration of SAX 2.0.2.
12:  *
13:  * @author  Computer Science E-259
14:  **/
15:
16: public class SAXDemo extends DefaultHandler
17: {
18:     /**
19:      * Main driver.  Expects one command-line argument:
20:      * the name of the file to parse.
21:      *
22:      * @param argv [0] - filename
23:      */
24:
25:     public static void main(String [] argv)
26:     {
27:         // ensure proper usage
28:         if (argv.length != 1)
29:         {
30:             System.out.println("Usage: java SAXDemo filename");
31:             System.exit(1);
32:         }
33:
34:         // grab filename
35:         String input = argv[0];
36:
37:         try
38:         {
39:
40:             // instantiate a SAX parser
41:             SAXParserFactory factory = SAXParserFactory.newInstance();
42:             SAXParser parser = factory.newSAXParser();
43:
44:             // instantiate our little demo handler
45:             SAXDemo handler = new SAXDemo();
46:
47:             // parse the file
48:             parser.parse(input, handler);
49:
50:         }
51:         catch (Exception e)
52:         {
53:             e.printStackTrace();
54:         }
55:     }
56:
57:
58:     /**
59:      * Report a startElement event.
60:      *
61:      * @param   uri namespace
62:      * @param   localName   name of element, sans namespace
63:      * @param   qName       name of element, with namespace
64:      * @param   attributes  element's collection of attributes
```

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

```
129:
130:     /**
131:      * Report an endDocument event.
132:      *
133:      * @throws  SAXException    general SAX error or warning
134:      */
135:
136:     public void endDocument() throws SAXException
137:     {
138:         System.out.println("endDocument();\n");
139:     }
140: }
```

```
 1: package cscie259.project1.mf;
 2:
 3: import java.io.DataInputStream;
 4: import java.io.File;
 5: import java.io.FileInputStream;
 6: import java.io.IOException;
 7:
 8:
 9: /**
10:  * A simplified XML parser.  In essence, this class supports a subset
11:  * of the functionality collectively offered by javax.xml.parsers.SAXParser
12:  * and javax.xml.parsers.DocumentBuilder.
13:  *
14:  * You MAY modify this file.
15:  *
16:  * @author  Computer Science E-259
17:  * @version 6.0
18:  *
19:  * @author  YOUR NAME GOES HERE
20:  **/
21: public class XMLParser
22: {
23:     /**
24:      * Storage for input file's contents.
25:      */
26:     private String data_;
27:
28:
29:     /**
30:      * A reference to the currently registered ContentHandler.
31:      */
32:     private ContentHandler handler_;
33:
34:
35:     /**
36:      * Index of our current location in input file's contents.
37:      */
38:     private int index_ = 0;
39:
40:
41:     /**
42:      * Returns true if the next characters in the stream are the beginning
43:      * of an element's end tag.
44:      *
45:      * @return true iff next characters in the stream are the beginning
46:      * of an element's end tag
47:      */
48:     protected boolean isEndTag()
49:     {
50:         return (data_.charAt(index_) == '<')
51:         && (data_.charAt(index_ + 1) == '/');
52:     }
53:
54:
55:     /**
56:      * Returns true if the next character in the stream is the beginning
57:      * of an element's start tag.
58:      *
59:      * @return true iff next character in the stream is the beginning
60:      * of an element's start tag
61:      */
62:     protected boolean isStartTag()
63:     {
64:         return data_.charAt(index_) == '<';
```

```
    }


    /**
     * Parses the specified file, if possible, passing SAX events
     * to given handler.
     *
     * @param filename name of file whose contents are to be parsed
     * @param handler  ContentHandler for SAX events
     */
    public void parse(String filename, ContentHandler handler)
    {
        // initialize to clean up from any previous parse
        data_ = "";
        index_ = 0;
        handler_ = handler;

        // attempt to open file and read contents into local storage
        try
        {
            File f = new File(filename);
            int filesize = (int) f.length();
            byte[] filebytes = new byte[filesize];
            DataInputStream in = new DataInputStream(new FileInputStream(f));
            in.readFully(filebytes);
            in.close();
            data_ = new String(filebytes);
        }
        catch (IOException E)
        {
            System.err.println("Error reading file: " + E.getMessage());

            return;
        }

        // parse the document; hopefully there's a root element!
        handler_.startDocument();
        readElement();
        handler_.endDocument();
    }


    /**
     * Parses an element and its content.
     */
    protected void readElement()
    {
        if (!isStartTag())
        {
            throw new RuntimeException(
                "Error: expecting the start of " + "a new element");
        }

        // parse end tag
        String name = readStartTag();

        // keep reading in more elements and text until an end tag
        // is encountered
        while (!isEndTag())
        {
            if (isStartTag())
            {
                readElement();
            }
```

```
129:             else
130:             {
131:                 readText();
132:             }
133:         }
134:
135:         // parse end tag, ensuring it matches most current start tag
136:         readEndTag(name);
137:     }
138:
139:
140:     /**
141:      * Parses an end tag, ensuring its name matches currently opened
142:      * element's name.
143:      *
144:      * @param checkName currently opened element's name with which
145:      * end tag should be compared
146:      */
147:     protected void readEndTag(String checkName)
148:     {
149:         // start name from scratch
150:         String name = "";
151:
152:         // read starting <
153:         index_++;
154:
155:         // read /
156:         index_++;
157:
158:         // read name
159:         while (data_.charAt(index_) != '>')
160:         {
161:             name += data_.charAt(index_);
162:             index_++;
163:         }
164:
165:         // read ending >
166:         index_++;
167:
168:         // ensure content is well-formed
169:         if (!checkName.equals(name))
170:         {
171:             throw new RuntimeException(
172:                 "Error: expecting closing tag for " + checkName);
173:         }
174:
175:         // pass this SAX event to handler
176:         handler_.endElement(name);
177:     }
178:
179:
180:     /**
181:      * Parses a start tag, returning opened element's name.
182:      *
183:      * @return name of element
184:      */
185:     protected String readStartTag()
186:     {
187:         // start name from scratch
188:         String name = "";
189:
190:         // Read starting <
191:         index_++;
192:
```

```
193:         // Read name
194:         while (data_.charAt(index_) != '>')
195:         {
196:             name += data_.charAt(index_);
197:             index_++;
198:         }
199:
200:         // Read ending >
201:         index_++;
202:
203:         // pass this SAX event to handler;
204:         // you MUST replace null below with a reference to
205:         // this element's Attributes object
206:         handler_.startElement(name, null);
207:
208:         // return this element's name, for later comparision
209:         // with an end tag
210:         return name;
211:     }
212:
213:
214:     /**
215:      * Parses character data.
216:      */
217:     protected void readText()
218:     {
219:         // start character data from scratch
220:         String content = "";
221:
222:         // accumulate characters until next tag
223:         while (data_.charAt(index_) != '<')
224:         {
225:             content += data_.charAt(index_);
226:             index_++;
227:         }
228:
229:         // pass this SAX event to handler
230:         handler_.characters(content);
231:     }
232: }
```