Acceso a ficheros XML con SAX

Introducción

- SAX (API Simple para XML) es un conjunto de clases e interfaces que ofrecen una herramienta muy útil para el procesamiento de documentos XML. Permite analizar documentos de forma secuencial (es decir, no carga todo el fichero en memoria como hace DOM), esto implica:
  - Poco consumo de memoria aunque los documentos sean de gran tamaño
  - Impide tener una visión global del documento
  - El acceso es secuencial NO aleatorio
  - SAX NO permite generar archivos XML
- Por tanto, SAX es útil cuando queremos buscar información en ficheros grandes XML o cuando no tenemos acceso completo al documento (acceso mediante *Stream*)
- SAX es una API totalmente escrita en Java e incluida dentro del JRE que nos permite crear nuestro propio parser de XML

Lectura de XML

- La lectura de un documento XML produce eventos que ocasiona la llamada a métodos:

Documento XML (alumnos.xml)	**Métodos asociados a eventos del documento**
<?xml version=”1.0”?>	startDocument()
<listadealumnos>	startElement()
<alumno>	startElement()
<nombre>	startElement()
Juan	characters()
</nombre>	endElement()
<edad>	startElement()
19	characters()

</edad>	endElement()
</alumno>	endElement()
<alumno>	startElement()
<nombre>	startElement()
María	characters()
</nombre>	endElement()
<edad>	startElement()
20	characters()
</edad>	endElement()
</alumno>	endElement()
</listadealumnos>	endElement()
	endDocument()

- Como se observa en la tabla:
  - La etiqueta de inicio (<?xml versión = “1.0”?>) → provoca el evento **startDocument()**
  - El final de document → provoca el evento **endDocument()**
  - La etiqueta de inicio de un elemento → provoca el evento **startElement()**
  - La etiqueta de final de un elemento → provoca el evento **endElement()**
  - Los caracteres entre etiquetas → provocan el evento **characters()**
  - …
- Los objetos que poseen los métodos que tratarán los eventos son:
  - **ContentHandler** → recibe las notificaciones de los eventos que ocurren en el documento
  - **DTDHandler** → recoge eventos relacionados con la DTD (Definición de Tipo de Documento)
  - **ErrorHandler** → define métodos de tratamientos de errores
  - **EntityResolver** → sus métodos se llaman cada vez que se encuentra una referencia a una entidad
  - **DefaultHandler** → clase que provee una implementación por defecto para todos sus métodos, el programador definirá los métodos que serán utilizados por el programa. Esta clase es la que extenderemos para poder crear nuestro parser de XML. En el ejemplo que veremos a continuación, la clase se llama *GestionContenido* y se tratan solo los eventos básicos: inicio y fin de documento, inicio y fin de etiqueta encontrada, encuentra datos carácter.

Ejemplo lectura de XML

```
import java.io.*;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;
```

Clases e interfaces de SAX

```
public class PruebaSax1 {
        public static void main (String [] args)
                throws FileNotFoundException, IOException, SAXException {
/* A continuación se crea objeto procesador XML - XMLReader -. Durante la creación de este objeto se puede producir una excepción SAXException. */
        XMLReader procesadorXML = XMLReaderFactory.createXMLReader();
/* A continuación, mediante setContentHandler establecemos que la clase que gestiona los eventos provocados por la lectura del XML será GestionContenido */
        GestionContenido gestor = new GestionContenido();
        procesadorXML.setContentHandler(gestor);
/* Por último, se define el fichero que se va leer mediante InputSource y se procesa el documento XML mediante el método parse() de XMLReader */
        InputSource fileXML = new InputSource ("alumnos.xml");
        procesadorXML.parse(fileXML);
        }
}
```

```
/* GestionContenido es la clase que implementa los métodos necesarios para crear nuestro parser de XML. Es decir, definimos los métodos que serán llamados al provocarse los eventos comentados anteriormente: startDocument, startElement, characters, etc. Si quisieramos tratar más eventos definiríamos el método asociado en esta clase. */

class GestionContenido extends DefaultHandler {
	public GestionContenido(){
		super();
	}
	public void startDocument(){
		System.out.println("Comienzo del documento XML");
	}
	public void endDocument(){
		System.out.println("Final del documento XML");
	}
	public void startElement (String uri, String nombre, String nombreC, Attributes atts) {
		System.out.printf("\tPrincipio Elemento: %s %n", nombre);
	}
	public void endElement (String uri, String nombre, String nombreC){
		System.out.printf("\tFin Elemento: %s %n",nombre);
	}
	public void characters(char[] ch, int inicio, int longitud) throws SAXException {
		String car = new String (ch, inicio, longitud);
		car = car.replaceAll("[\t\n]","");
		System.out.printf("\tCaracteres: %s %n", car);
	}
}
```

Problemas AccesoSAX

1. Copia el ejemplo anterior y ejecútalo.
2. Prueba a leer con el programa del ejercicio anterior el documento “personas.xml” (se trata del xml que generamos en la clase anterior)
3. Crea un programa que lea el archivo “discoteca.xml” (adjunto en el moodle) y determine el número de discos listados en el archivo. AYUDA: contabilizando el número de títulos.
4. Crea un programa que devuelva el número de discos registrados en “discoteca.xml” de un determinado autor que le pasamos por consola. Si el autor carece de discos en el archivo, el programa devolverá un mensaje del estilo: “El autor <xxxxxx> no aparece en el archivo.