```
    //Para la creación del parser
    DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();

  try {
      builder = factory.newDocumentBuilder();//obtenemos un procesador o parser XML
      document = builder.parse(fileIn); //cargamos el documento desde fileIn(Empleados.xml)
      document.getDocumentElement().normalize(); //obtiene los elementos permitiendo su acceso
      //imprime el nombre del nodo raíz
      System.out.println("Elemento raíz: " + document.getDocumentElement().getNodeName());

      //crea una lista de nodos con todos los nodos empleado
      empleados = document.getElementsByTagName("empleado");

      //recorre la lista de nodos
      for (int i = 0; i < empleados.getLength(); i++) {
        emple = empleados.item(i); //obtiene un nodo
        // si es un nodo tipo elemento
        if (emple.getNodeType() == Node.ELEMENT_NODE) {
          Element elemento = (Element) emple; //obtiene los elementos del nodo
          System.out.println("ID: " + getNodo("id", elemento));
          System.out.println("Apellido: " + getNodo("apellido", elemento));
          System.out.println("Departamento: " + getNodo("dep", elemento));
          System.out.println("Salario: " + getNodo("salario", elemento));
        }
      }
    } catch (ParserConfigurationException ex) {
      //puede provocarla '.newDocumentBuilder()'
      System.out.println("Error de construcción del Lector");
    } catch (IOException ex) {
      //puede provocarla '.parse()'
      System.out.println("Error de acceso al fichero origen");
    } catch (SAXException ex) {
      //también puede provocarla '.parse()'
      System.out.println("Error de conversión del Lector");
    }
  }//fin de main

  /** Método para obtener la información de un nodo*/
  private static String getNodo(String etiqueta, Element elem) {

    NodeList nodo = elem.getElementsByTagName(etiqueta).item(0).getChildNodes();
    Node valornodo = (Node) nodo.item(0);
    return valornodo.getNodeValue();//devuelve el valor del nodo
  }
}//fin de la clase
```

7.2 ACCESO A FICHEROS XML CON SAX

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 los documentos de forma secuencial (es decir, no carga en memoria todo el fichero como hace DOM), esto implica poco consumo de memoria aunque los documentos sean de gran tamaño, en contraposición, impide tener una visión global del documento que se va a analizar.
- SAX es más complejo de programar que DOM.
- Es una API totalmente escrita en Java e incluida dentro del JRE que nos permite crear nuestro propio parser de XML.

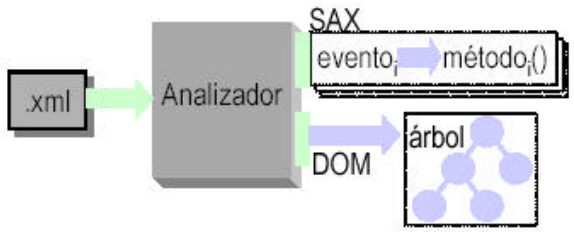
La **lectura de un documento XML** produce eventos que ocasiona la llamada a métodos. Los eventos son encontrar:

- la etiqueta de inicio y fin del documento (startDocument() y endDocument())
- la etiqueta de inicio y fin de un elemento (startElement() y endElement())
- los caracteres entre etiquetas (characters()),
- etc.

Documento XML (Alumnos.xml)	Métodos asociados a eventos del documento
<?xml version="1.0"?> <listadealumnos> <alumno> <nombre> Juan </nombre> <edad> 19 </edad> </alumno> <alumno> <nombre> Maria </nombre> <edad> 20 </edad> </alumno> </listadealumnos>	startDocument() startElement() startElement() startElement() characters() endElement() startElement() characters() endElement() endElement() startElement() startElement() characters() endElement() startElement() characters() endElement() endElement() endDocument()

A medida que el analizador XML va leyendo el documento xml (Alumnos.xml) y encuentra (los eventos), los componentes del documento (elementos, atributos, valores, etc), o detecta errores, va invocando a las funciones que ha asociado el programador (métodos que ha sobreescrito).

El siguiente dibujo muestra los enfoques de DOM y SAX como analizadores de documentos XML.



La API necesaria para trabajar con SAX es **org.xml.sax**

Vamos a construir un ejemplo sencillo en Java que muestra los pasos báscios necesarios para utilizar SAX

1. Las clases necesarias de SAX en el ejemplo son:

```
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;
```

2. Se crea un objeto procesador o analizador de XML, un objeto **XMLReader.** La creación de este objeto puede producir la excepción *SAXException*, que es necesario capturar.

```
XMLReader procesadorXML = XMLReaderFactory.createXMLReader();
```

3. Hay que indicar al ***XMLReader*** qué objetos poseen los métodos que tratarán los eventos. Para este fin, existen cuatro interfaces con métodos que se pueden asociar a los eventos:
   - *ContentHandler*: eventos sobre datos (el principal y el más extenso)
   - *DTDHandler*: recoge eventos relacionados con la DTD.
   - *ErrorHandler*: define métodos de tratamiento 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 sean utilizados por el programa.

En este ejemplo, la clase ***DefaultHandler*** es de la que extenderemos para poder crear y gestionar el analizador XML.

```
class GestionContenido extends DefaultHandler
```

En el ejemplo la clase se llama *GestionContenido* y se tratan solo los eventos básicos: inicio y fin de documento, inicio y fin de etiqueta encontrada, y encuentra datos carácter; por lo que los **métodos a sobreescribir** serán:

- ***startDocument():*** se produce al comenzar el procesado del documento XML.
- ***endDocument()***: se produce al finalziar el procesado del documento XML.
- ***startElement()***: se produce al comenzar el procesado de una etiqueta XML. Es aquí donde se leen los atributos de las etiquetas.
- ***endElement():*** se produce al finalizar el procesado de una etiqueta XML
- ***characters()***: se produce al encontrar una cadena de texto.

Dependiando de la interfaz utilizada, habrá que indicar al procesador XML los objetos que realizarán el tratamiento, mediante alguno de los siguientes métodos: *setContentHandler(), setDTDHandler(), setEntityResolver(), setErrorHandler() (*cada uno trata un tipo de evento y está asociado con una interfaz determinada).

En el ejemplo usaremos ***setContentHandler()*** para tratar los eventos que ocurren en el documento:

```
GestionContenido gestor = new GestionContenido();
procesadorXML.setContentHandler(gestor);
```

4. A continuación se define el fichero XML que se va a leer mediante un objeto *InputSource*:

```
InputSource fileXML = new InputSource("c:\\ad\\ud2\\alumnos.xml");
```

5. Por último se procesa el documento XML mediante el método *parse()* de *XMLReader*, y le pasamos un objeto *InputSource*:

```
procesadorXML.parse(fileXML);
```

EJEMPLO 19. Código completo del ejemplo que realiza la lectura del documento XML con los datos de alumnos Alumnos.xml. [PruebaSax1.java]

```
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;

public class PruebaSax1 {

  public static void main(String[] args) {

    XMLReader procesadorXML;

    try {
      //crea un objeto procesador o analizador de XML
      procesadorXML = XMLReaderFactory.createXMLReader();
      //Clase que extiende a DefaultHandler para la gestión del analizador XML
      GestionContenido gestor = new GestionContenido();
      procesadorXML.setContentHandler(gestor);
      //Fichero con documento xml
      InputSource fileXML = new InputSource("c:\\ad\\ud2\\alumnos.xml");
      //se procesa el documento XML
      procesadorXML.parse(fileXML);

    } catch (SAXException ex) {
      //puede provocarla '.createXMLReader()'
      System.out.println("No se pudo crear el procesador XML");
```

```
    } catch (IOException ex) {
      //puede provocarla '.parse()'
      System.out.println("Error de acceso al fichero origen");
    }

  }
}//fin PruebaSaxl

//clase de la que extendemos para gestionar el analizador XML
class GestionContenido extends DefaultHandler {
  public GestionContenido() {
    super();
  }
//métodos que sobreescribimos para nuestro tratamiento
   @Override
  public void startDocument() {
    System.out.println("Comienzo del Documento XML");
  }
   @Override
  public void endDocument() {
    System.out.println("Final del Documento XML");
  }
  @Override
  public void startElement(String uri, String nombre,
          String nombreC, Attributes atts) {

    System.out.println("\tPrincipio Elemento: " + nombre);
  }
  @Override
  public void endElement(String uri, String nombre, String nombreC) {
    System.out.println("\tFin Elemento: " + nombre);
  }
   @Override
  public void characters(char[] ch, int inicio, int longitud)
          throws SAXException {

    String car = new String(ch, inicio, longitud);
    car = car.replaceAll("[\t\n]", "");//quitar saltos de linea y tabulador
    System.out.println("\tCaracteres: " + car);
  }
}//fin GestionContenido
```

7.3 ANALIZADORES XML Y VALIDACIÓN

Los analizadores permiten seleccionar diferentes opciones, entre ellas si ha de comprobarse o no la validez de un documento xml respecto a su definición (el DTD). Esta validación añade una comprobación y por tanto una garantía más, pero también consume más recursos y ralentiza el procesamiento del documento xml.