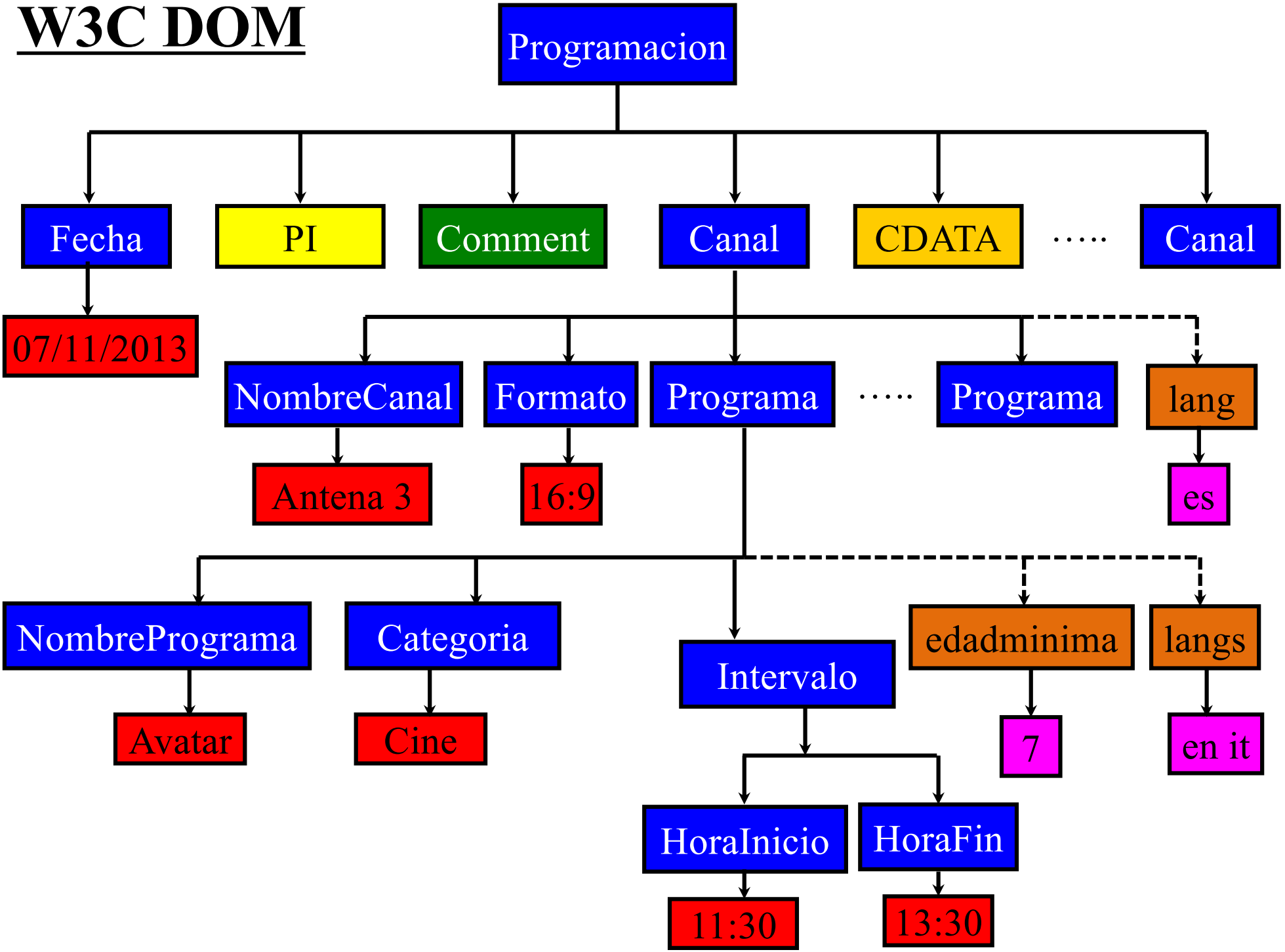
TVML: Cartelera de televisión

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Programacion>
    <Fecha>07/11/2013</Fecha>
    <?xml-stylesheet href="estilos.css"> <!-- en caso de que haya que presentarlo -->

    <Canal lang="es">
        <NombreCanal >Antena 3</NombreCanal>
        <Formato> 16:9 </Formato>
        <Programa edadminima="7" langs="en it">
            <NombrePrograma>Avatar</NombrePrograma> <Categoria>Cine</Categoria>
            <Intervalo><HoraInicio>11:30</HoraInicio><HoraFin>13:30 </HoraFin></Intervalo>
        </Programa>
        ……
    </Canal>
    <![CDATA[ Este contenido no se analiza, se pasa tal cual a la aplicación ]]>

    <Canal lang="es">
    ……
    </Canal>
    ……
</Programacion>
```

W3C DOM

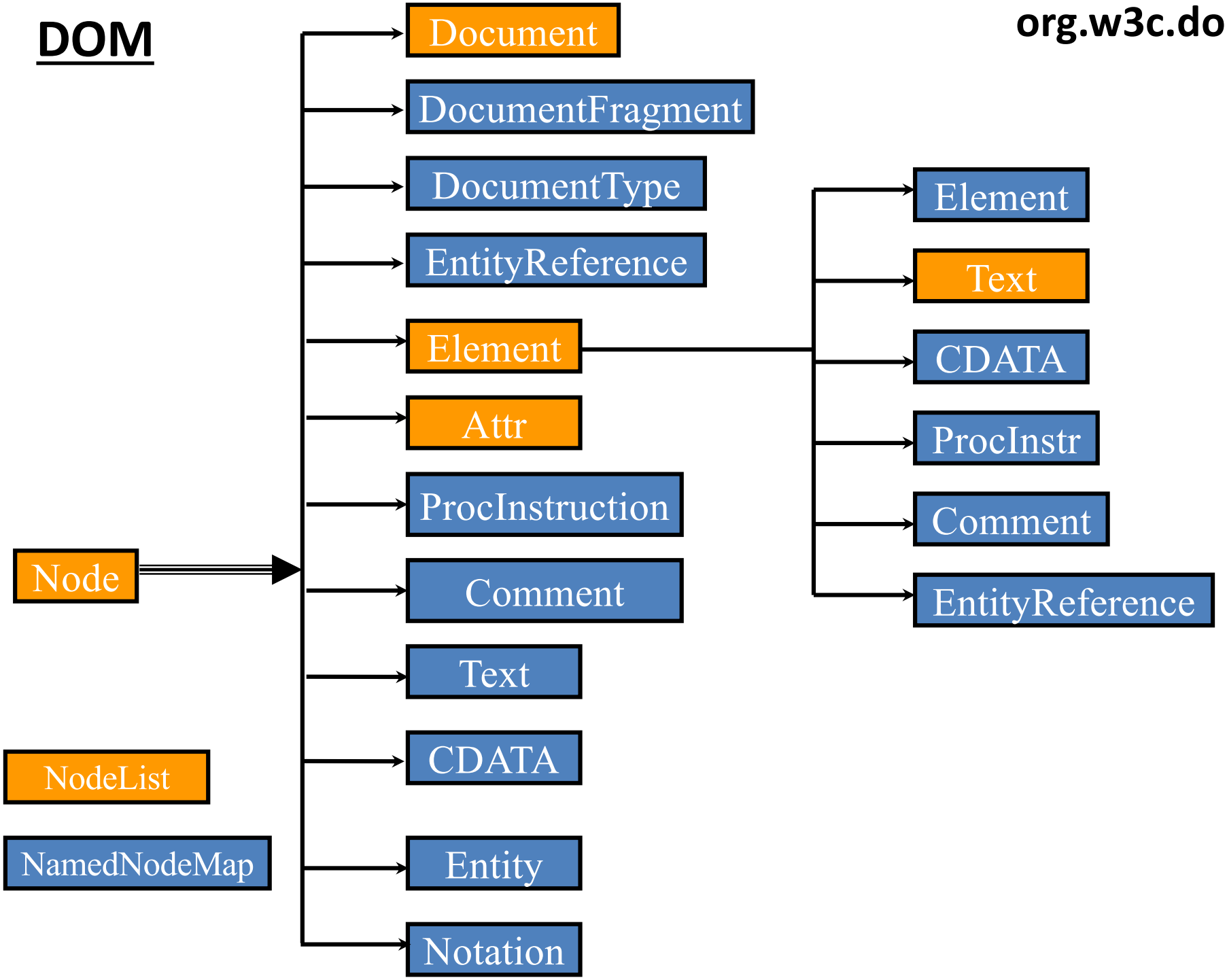

Programacion
Fecha
PI
Comment
Canal
CDATA
.....
Canal
07/11/2013
NombreCanal
Formato
Programa
.....
Programa
lang
Antena 3
16:9
es
NombrePrograma
Categoria
Intervalo
edadminima
langs
Avatar
Cine
7
en it
HoraInicio
HoraFin
11:30
13:30

API JAXP

- API para comprobar la corrección de documentos XML, obtener información de ellos (a través de su árbol DOM), e incluso modificarlos
- Incluida por defecto en la distribución Java
- Compuesta por varios paquetes, entre ellos:
  - javax.xml.parsers → paquetes para crear, configurar y usar *parsers*
    - clases DocumentBuilderFactory, DocumentBuilder
  - org.w3c.dom → interfaz DOM para acceder al documento
    - clases Document, Node, NodeList, Element…
    - Constantes org.w3c.dom.Node.TEXT_NODE….
  - org.xml.sax → interfaz de parsers SAX (define excepciones que usan algunos métodos de los interfaces anteriores)
  - org.xml.sax.helpers → clases auxiliares
    - DefaultHandler (para ser notificados de los posibles errores del análisis)

DOM

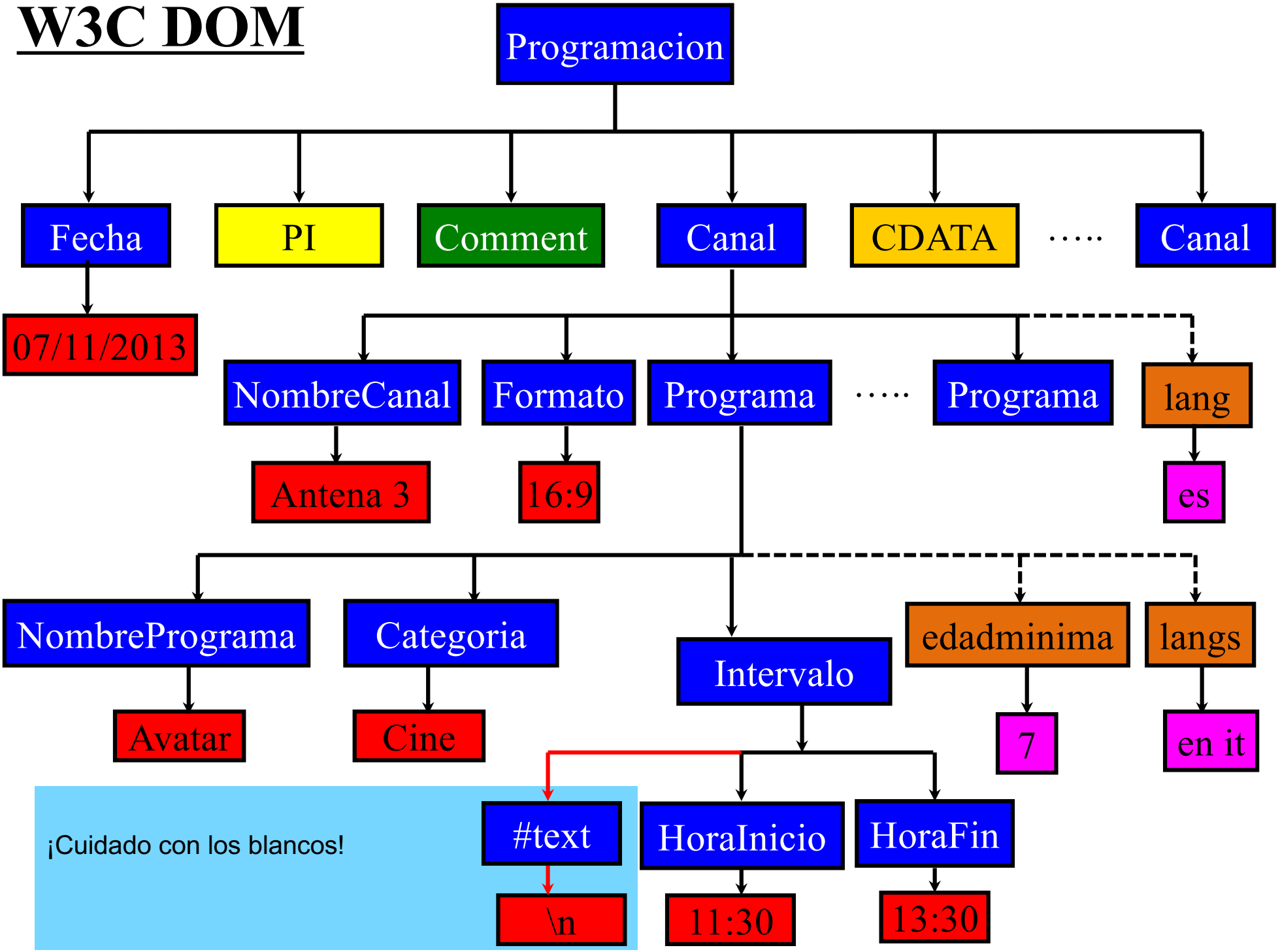
org.w3c.dom



JAXP: lectura de un documento sin un Schema

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.DocumentBuilder;
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

public class TVML_Parser {
   public static void main (String[] args)   {
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();

        try {
           Document doc = db.parse(URL o fichero);
        }
        catch (SAXException e)  { // código }     // provocada por documento no well-formed

        Element programacion = doc.getDocumentElement();      // elemento raíz


  }
}     // puede faltar capturar otras excepciones
```

W3C DOM


Programacion
Fecha
PI
Comment
Canal
CDATA
.....
Canal
07/11/2013
NombreCanal
Formato
Programa
.....
Programa
lang
Antena 3
16:9
es
NombrePrograma
Categoria
Intervalo
edadminima
langs
Avatar
Cine
7
en it
#text
HoraInicio
HoraFin
\n
11:30
13:30
¡Cuidado con los blancos!

Document Object Model (DOM)

- **Node**:
  - **Propiedades:** `nodeName`, `nodeValue`, `nodeType` (CONSTANTES, p.e. `org.w3c.dom.Node.TEXT_NODE`), `parentNode`, `childNodes`, etc.
  - **Métodos:** `getNodeName()`, `getNodeType()`, `getNodeValue()`, `getChildNodes()`, `getFirstChild()`, `insertBefore()`, `removeChild()`, etc.

- **Document**: `getDoctype()`, `getDocumentElement()`, `getElementsByTagName()`, `createAttribute()`, `createElement()`

- **Element**: `getAttribute()`, `getElementsByTagName()`, `setAttribute()`, `appendChild()`, etc.

- **Attr:** `getName()`, `getValue()`, etc.

nodeType
DOCUMENT_NODE
ELEMENT_NODE
ATTRIBUTE_NODE
TEXT_NODE
…

- TODAS LAS CLASES, PROPIEDADES Y MÉTODOS ESTÁN DESCRITOS EN EL JAVADOC DE LA API QUE SE USE, EN NUESTRO CASO JAXP
  - El **nodeName** de elementos y atributos es su nombre. El de **Text** es #text
  - El **nodeValue** de **Text** y **Attr** es su contenido. El de los elementos es indefinido

Ejemplo: buscar información en TVML

```
<Programacion>
   <Fecha>20/12/2004</Fecha>

   <Canal lang='es' teletexto='si'>
      <NombreCanal>TVE1</NombreCanal>
      <Numero>1</Numero>

      <Programa edadminima='3' langs='es'>
         <NombrePrograma>Barrio Sésamo</NombrePrograma>
Un comentario
         <Categoria>Infantiles</Categoria>
         <HoraInicio>17:00</HoraInicio>
         <HoraFin>18:00</HoraFin>
      </Programa>
      …. <!– más programas →

   </Canal>

   …. <!– más canales →
</Programacion>
```

Supongamos que “e” es un nodo tipo Element

- String valor = **e**.getAttribute(“nombre_atributo”)
  - String lang = **elem_canal**.getAttribute(“lang”) // Element elem_canal;
- NodeList nl = **e**.getChildNodes()
  - NodeList nl_hijosdecanal = **elem_canal**.getChildNodes(); // no devuelve atributos
- Node nodo = **e**.getFirstChild()
  - Element elem_nombreCanal = (Element)**elem_canal**.getFirstChild() // peligroso, por blancos
- NodeList nl = **e**.getElementsByTagName(“nombre”)
  - NodeList nl_fechas = **elem_programacion**.getElementsByTagName(“Fecha”)
    - Element elem_fecha = (Element)nl_fechas.item(0)
  - NodeList nl_programas = **elem_canal**.getElementsByTagName(“Programa”)
    - Element elem_programa1 = (Element)nl_programas.item(0)
- String texto = **e**.getTextContent() // todo el texto agregado, suyo y de sus descendientes
  - String fecha = **elem_fecha**.getTextContent()
- ~~String fecha = **elem_fecha**.getNodeValue()~~ ¡¡¡NO!!!
  - Node txtNode= **elem_fecha**.getFirstChild()
  - String fecha = txtNode.getNodeValue()

Leer las URLs de los nuevos ficheros

- Suponiendo que **elem_programacion** es el nodo raíz:

```
NodeList nl_canales = elem_programacion.getElementsByTagName(“Canal”)
Element elem_canal1 = (Element) nl_canales.item(0)
NodeList nl_programas = elem_canal1.getElementsByTagName(“Programa”)
Element elem_programa1= (Element)nl_programas.item(0)
NodeList nl_TVMLs = elem_programa1.getElementsByTagName(“TVML”)
Element elem_TVML1 = (Element)nl_TVMLs.item(0)
Node txtNode= elem_TVML1.getFirstChild()    // peligroso por los blancos
String nuevaURL = txtNode.getNodeValue()
```

- Suponiendo que **doc** es el nodo documento:

```
NodeList nl_TVMLs = doc.getElementsByTagName(“TVML");
Element elem_TVML1 = (Element)nl_TVMLs.item(0);
String nuevaURL = elem_TVML1.getTextContent().trim();
```

Contenido mixto

- Cómo leer el texto de contenido mixto

```
<e1>                                  // queremos leer “hola”
      hola
      <e2> mundo </e2>          // e1.getTextContent()   devuelve “hola mundo”
</e1>
```

```
public String getComentarioPrograma (Element elemprograma) {
   NodeList nlHijos = elemPrograma.getChildNodes();
   String comentario = "";
   for (int x=0; x < nlHijos.getLength(); x++) {
       Node hijo = nlHijos.item(x);
       if (hijo.getNodeType() == org.w3c.dom.Node.TEXT_NODE)
            comentario = comentario+hijo.getNodeValue();  // puede haber varios
      }
      comentario = comentario.trim();
      return comentario;
}
```