EISTI
Ecole Internationale des Sciences du Traitement de l'Information

Traitements des données

JAXP : Java et XML

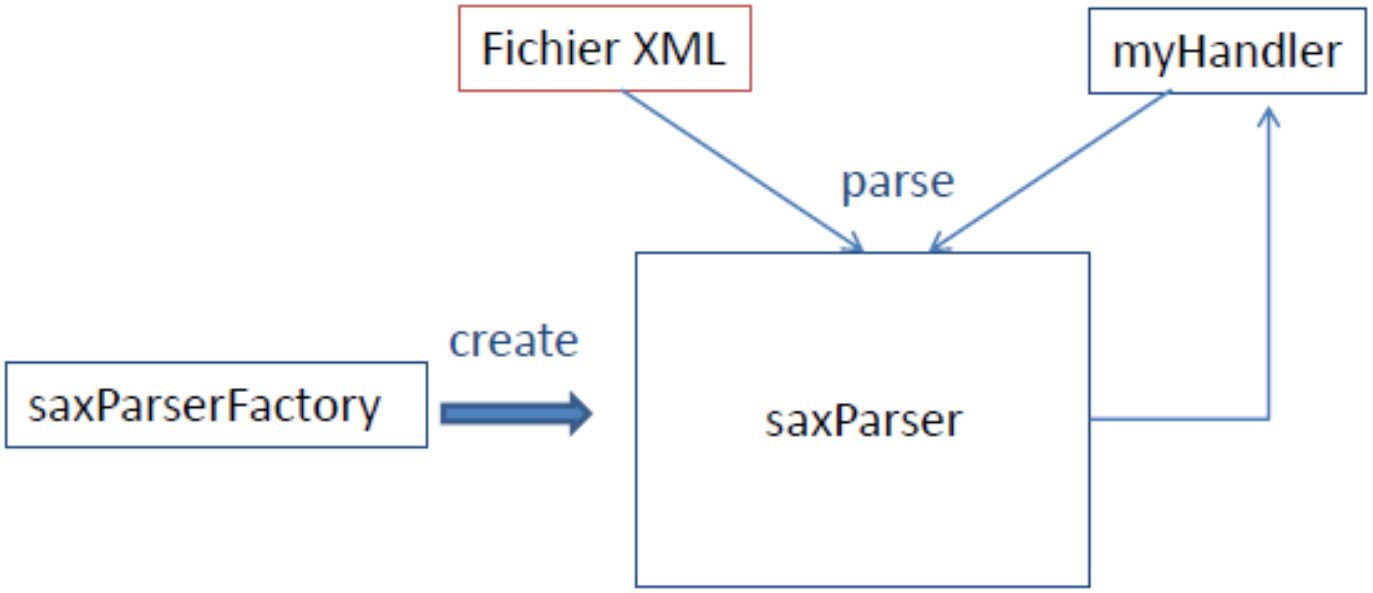
JAXP : Java API for XML Processing

- Interface de programmation Java de Sun permettant la création, la manipulation et le traitement de fichiers XML à bas niveau.
- Cette interface contient trois parties
  1. SAX : programmation évènementielle sur le parcours d'un arbre XML
  2. DOM : Création , modification et sauvegarde d'arbres XML
  3. XSLT : Transformation de fichiers XML à l'aide de feuilles XSL

Présentation de SAX

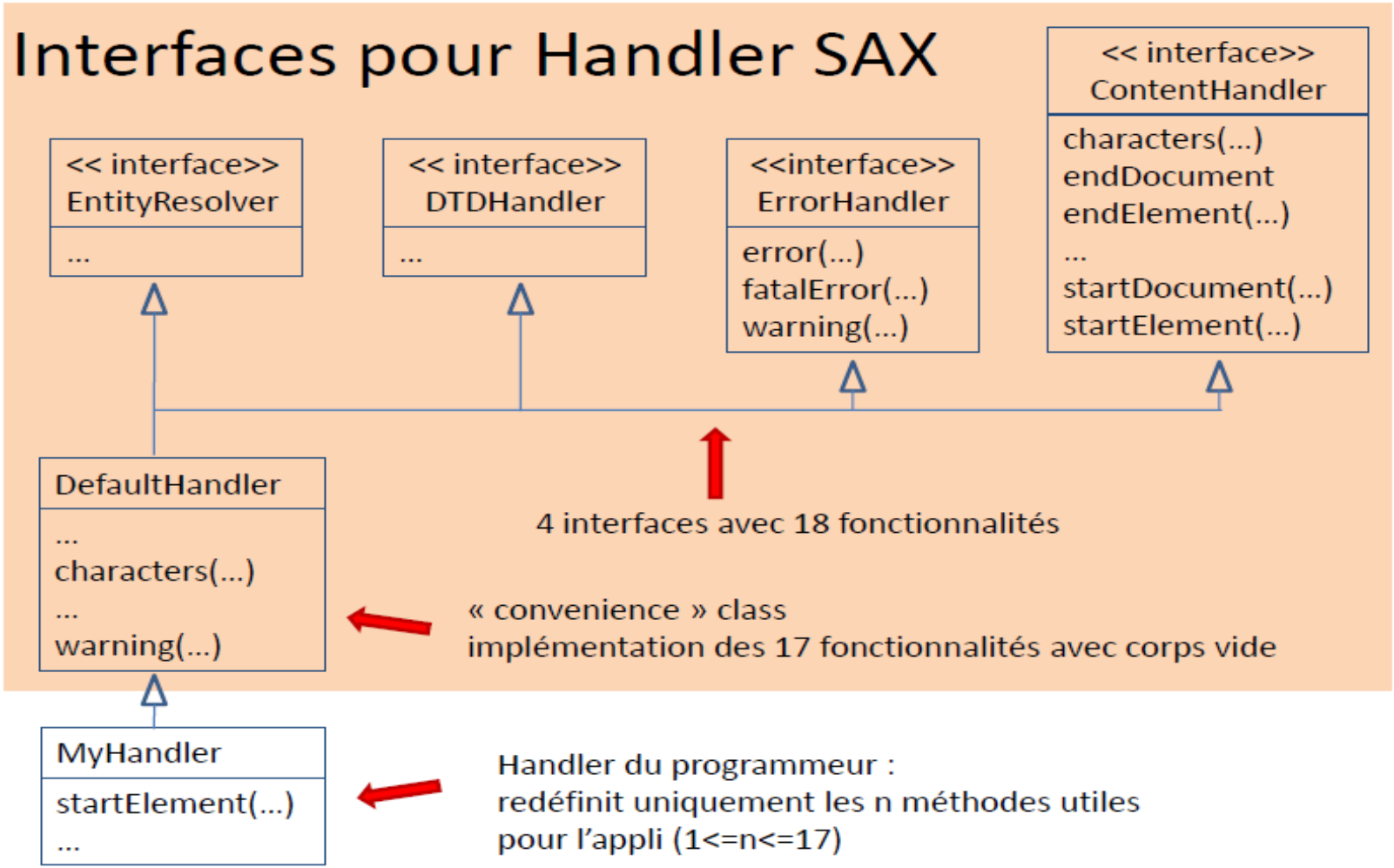
- SAX : Simple API for XML
- Site projet : http://www.saxproject.org
- Inclus JDK de Sun :
  - org.xml.sax
  - javax.xml.parsers
- Programmation évènementielle sur le parcours du document XML suivant le type de fragment rencontré (élément, texte, commentaire, …) => parser+ handlerSAX

SAX : Principe général

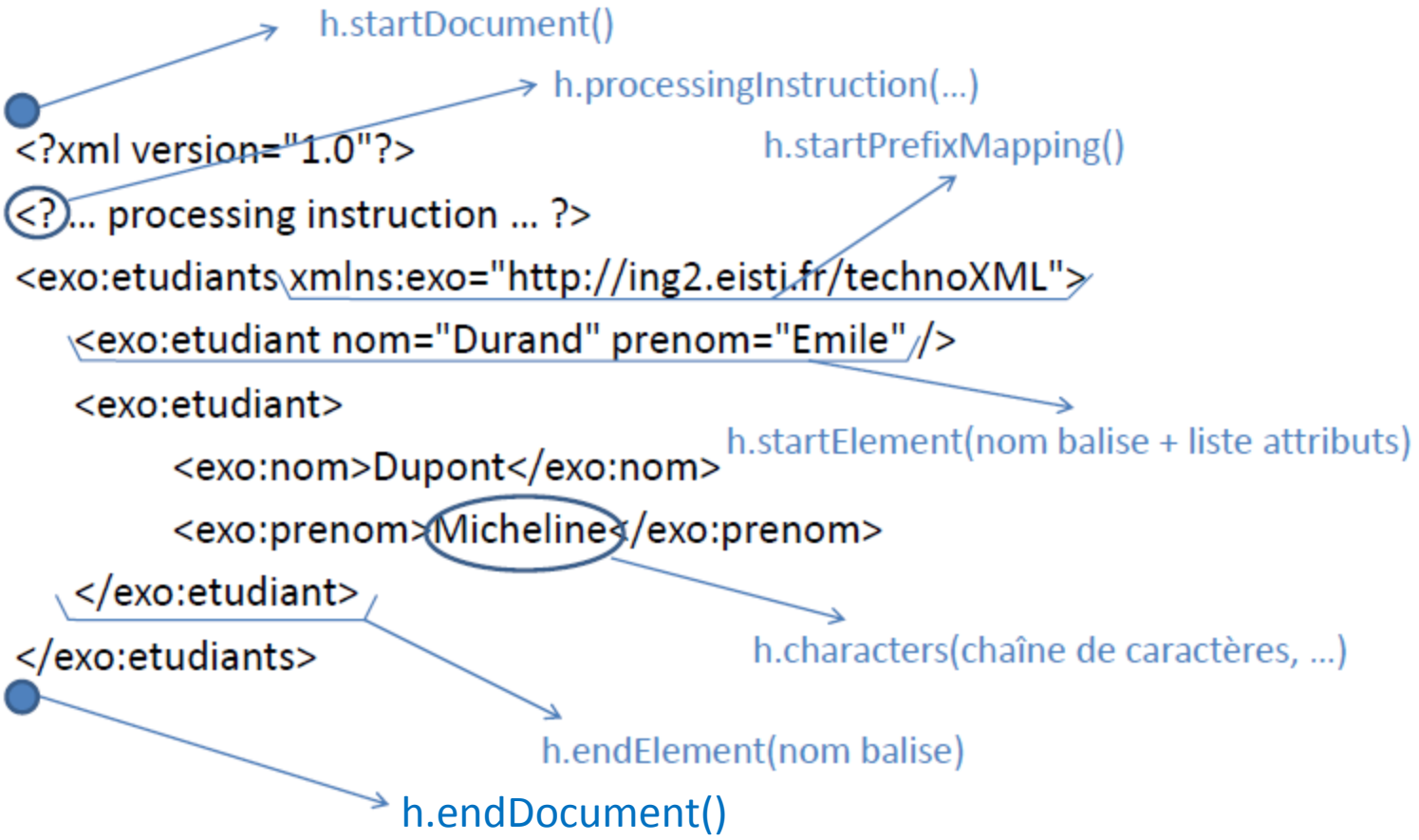


- classes SAXParser et SAXParserFactory (package javax.xml.parsers)
- fichier XML + handler fournis à la méthode parse
- Le parser appelle les méthodes appropriées du handler

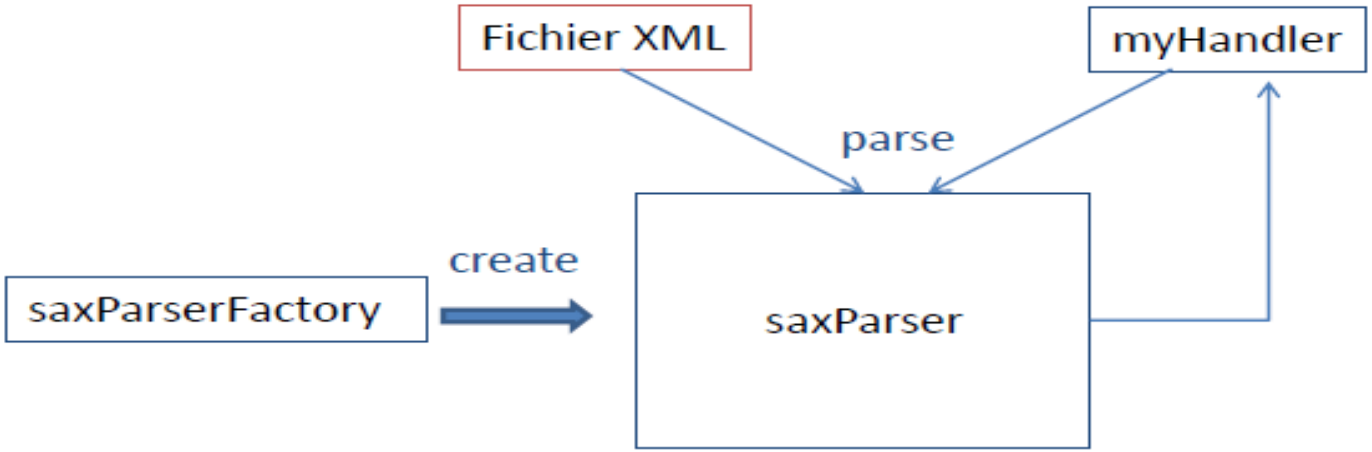
SAX : Interfaces du Handler



Exemple


h.startDocument()
h.processingInstruction(...)
h.startPrefixMapping()
<?xml version="1.0"?>
<? ... processing instruction ... ?>
<exo:etudiants xmlns:exo="http://ing2.eisti.fr/technoXML">
<exo:etudiant nom="Durand" prenom="Emile" />
<exo:etudiant>
h.startElement(nom balise + liste attributs)
<exo:nom>Dupont</exo:nom>
<exo:prenom>Micheline</exo:prenom>
</exo:etudiant>
h.characters(chaîne de caractères, ...)
</exo:etudiants>
h.endElement(nom balise)
h.endDocument()

Parsing avec SAX : L'objet myHandler



- La classe MyHandler doit dériver de la classe DefaultHandler.
- Elle doit contenir tous les attributs servant à l'analyse du parcours.
- Ces attributs sont mis à jour lors du parcours grace aux appels des traitements par l'objet saxParser.
- Une méthode publique doit être définie dans la classe MyHandler pour analyser les résultats après le parcours.
- Dans la suite, on appellera analyserResultats cette méthode.

Parsing avec SAX : Code Java

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;

public MaClasse {
public static void main (String args ) {
        DefaultHandler handler = new MyContentHandler(); // Classe d'evts définies par
        ailleurs
        SAXParserFactory factory = SAXParserFactory.newInstance();
        try
                {
                SAXParser saxParser = factory.newSAXParser();
                saxParser.parse( new File(???), handler); // ??? Le chemin d'accès au fichier xml
                // On analyse les résultats  du parsing
                handler.analyserResultats();
                }
        }
}
```

Parsing avec SAX : Validation XSD d'un XML

```
// On crée la fabrique de parser
SAXParserFactory spf = SAXParserFactory.newInstance();
spf.setNamespaceAware(true); // à mettre à vrai ou faux s'il y a ou pas des espaces de nommage
spf.setValidating(false);  // à mettre à vrai dans le cas d'une DTD et mettre à faux pour un XSD

// On charge le schéma XSD en mémoire vive et on l'attache à la fabrique de parser SAX
SchemaFactory sf = chemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema xsd = sf.newSchema(new File(xsdFilename));
spf.setSchema(xsd);    // on attache le schéma à la fabrique de parser

//  On crée le parser et il fait son travail
SAXParser sp = spf.newSAXParser();
MyHandler handler = new MyHandler(...);
 sp.parse(new File(xmlFilename), saxHandler);

// On analyse les résultats  du parsing
handler.analyserResultats();
```

L'interface ErrorHandler

- Parsage d'un document XML
  - Document XML mal formé
  - Document XML ne respectant pas une DTD ou un XSD

- Parsage d'un document XML avec SAX
  - **Erreur fatale** (*fatalError(SAXParseException exception) throws SAXException*) : Le parseur s'arrête (Ex : XML mal formé)
  - **Erreur** (*error(SAXParseException exception) throws SAXException*) : Le parseur peut continuer son travail (Ex : non respect DTD ou XSD)
  - **Warning** (*warning(SAXParseException exception) throws SAXException*) : Le parseur SAX continue le parsing du document avec un simple avertissement.

DOM : Document Object Model

- DOM est une normalisation du consortium W3C :

http://www.w3.org

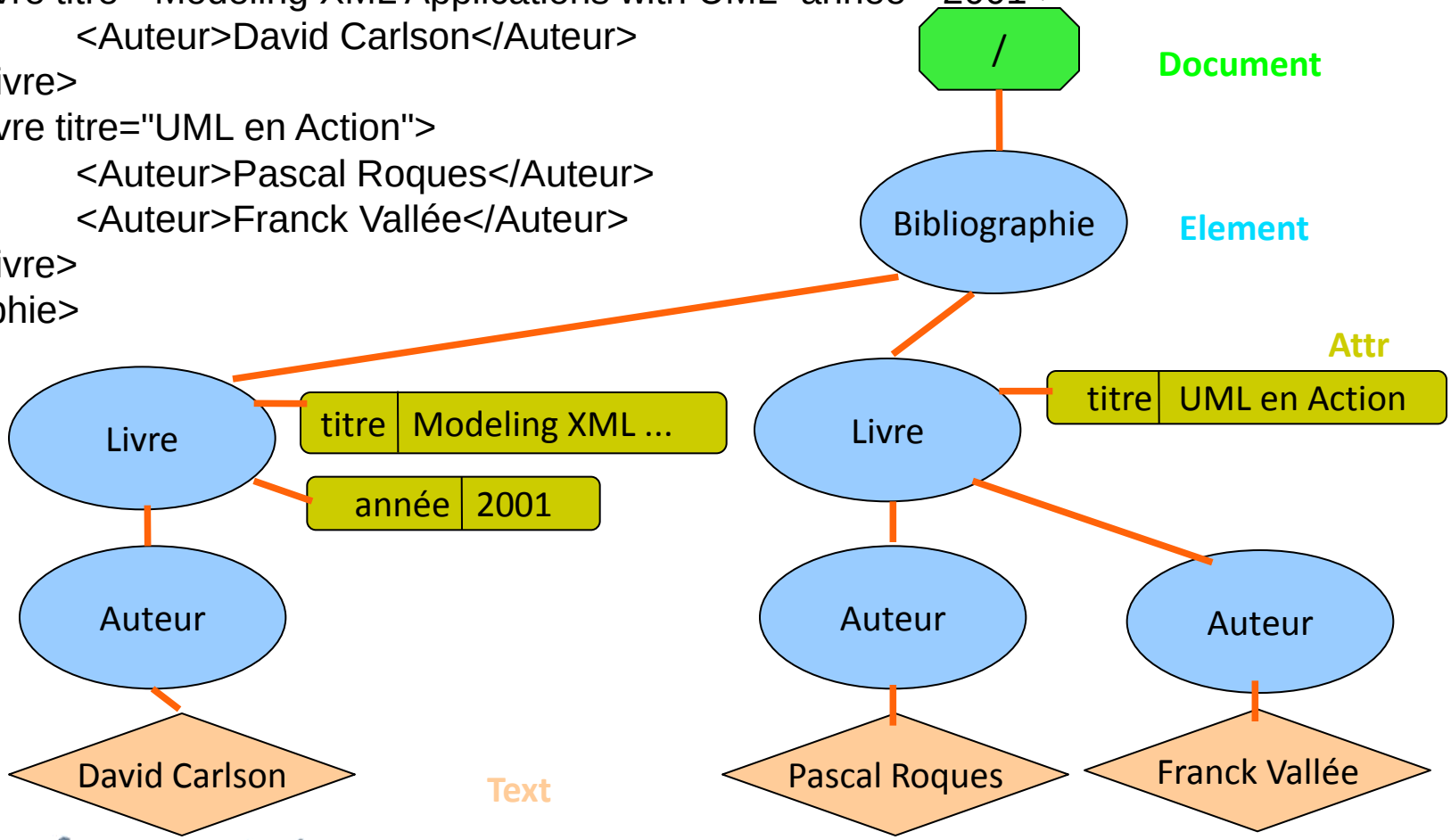
- DOM est implémenté dans de nombreux langages informatiques dont Java et Javascript.
- Dans le langage Java, le package org.w3c.org est inclus dans le JDK (il existe également une autre version opensource JDom)
- DOM est un ensemble d'interfaces qui permettent de :
  - représenter sous la forme d'un arbre un fichier XML
  - naviguer dans l'arbre
  - faire une recherche dans un arbre
  - modifier un arbre
  - sauvegarder un arbre XML

package org.w3c.dom

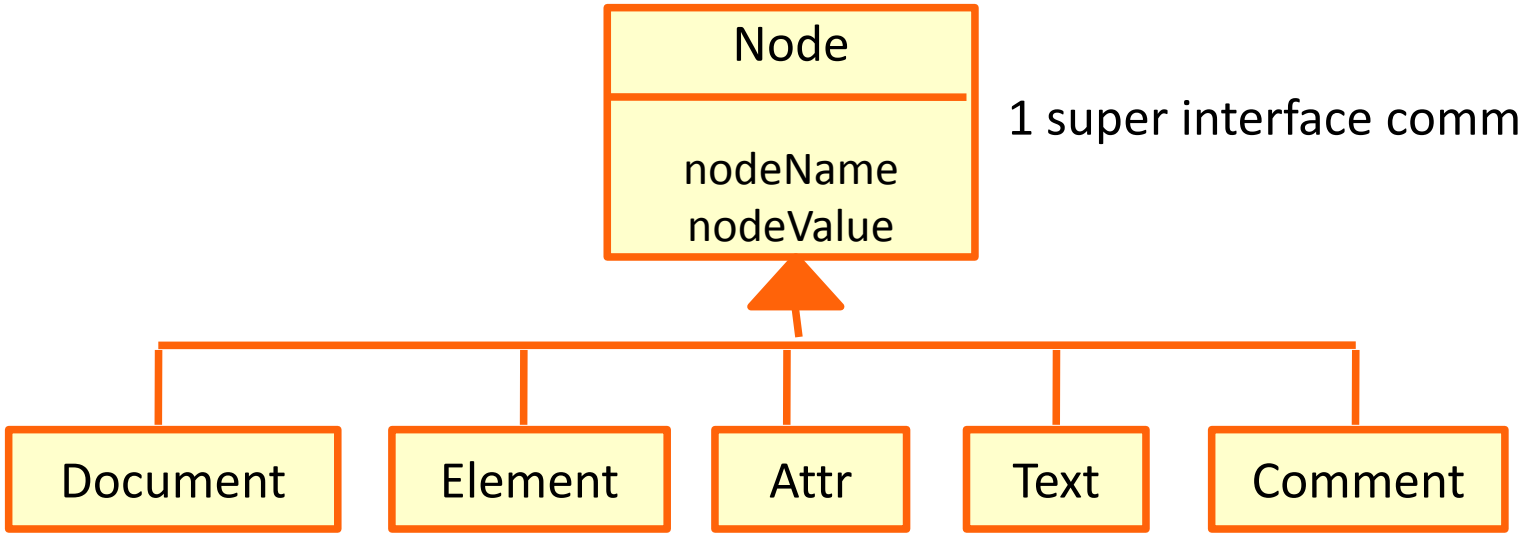
Interface	Partie de l'arbre
interface Document	Arbre XML
interface Element	Élément XML
interface Attr	Attribut d'un élément XML
interface Text	Texte d'un élément XML
interface Comment	Commentaire XML

Correspondance XML et Interface DOM

```
<Bibliographie>
        <Livre titre="Modeling XML Applications with UML "année="2001">
                <Auteur>David Carlson</Auteur>
        </Livre>
        <Livre titre="UML en Action">
                <Auteur>Pascal Roques</Auteur>
                <Auteur>Franck Vallée</Auteur>
        </Livre>
</Bibliographie>
```



DOM : Interfaces et attributs



1 super interface commune

	nodeName	nodeValue
Document	"#document"	null
Element	nom de balise	null
Attr	nom attribut	valeur attribut
Text	"#text"	texte
Comment	"#comment"	commentaire

DOM : Chargement d'un fichier XML


```
import org.w3c.dom.*;
import javax.xml.parsers.*; // DocumentBuilderFactory, DocumentFactory
public void chargerXML() {
        try {
        // création d'une fabrique de documents, fabrique de constructeurs
        DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();
        // création d'un constructeur de documents
        DocumentBuilder parser = fabrique.newDocumentBuilder();
        // lecture du contenu d'un fichier XML avec DOM
        File fdom = new File("???"); // ??? chemin du fichier xml
        Document dom = parser.parse(fdom);
        }
        catch(Exception e) { }
}
```

DOM : Navigation dans un objet Document

- Pour un document :
  - getDocumentElement() : élément racine de l'arbre

- Pour un élément :
  - getChildNodes() : liste des fils d'un élément de type
    - Element
    - Text
    - Comment
  - getAttributes() : liste des attributs d'un élément

- Pour un noeud quelconque :
  - getNodeName() : son nom (cf tableau)
  - getNodeValue() : sa valeur (cf tableau)
  - getNodeType() : son type (ELEMENT_NODE, TEXT_NODE, etc ...)

DOM : Recherche dans un objet Document

- A partir d'un document
  - getElementById : obtenir un élément par la valeur de son attribut ID unique
  - getElementsByTagName : obtenir une liste d'éléments par leur nom
  - getElementsByTagNameNS : obtenir une liste d'éléments par leur nom qualifié par un espace de nom (ex: xs:stylesheet)

- A partir d'un élément
  - getElementByTagName / getElementByTagNameNS : recherche uniquement à partir de cet élément
  - getAttribute / getAttributeNS : obtenir la valeur d'un attribut à partir de son nom
  - getAttributeNode / getAttributeNodeNS : obtenir un attribut à partir de son nom

DOM : Modification d'un objet Document

- Pour un noeud quelconque :
  - Remplacer la valeur du noeud : setNodeValue(nouvelleValeur)

- Pour un élément :
  - Supprimer un fils : removeChild(fils)
  - Remplacer un fils : replaceChild(nouveauFils, ancienFils)
  - Ajouter un attribut / Remplacer la valeur d'un attribut : setAttribute(nomAttribut, valeurAttribut)
  - Supprimer un attribut : removeAttribute(nomAttribut)

- Pour un document :
  - createElement / createTextNode / createComment => un noeud est toujours créé dans le contexte de son document, puis rattaché au bon endroit dans l'arbre avec les méthodes ci-dessus

JAXP : Transformation

Type de source
DomSource
SAXSource
StAXSource
StreamSource

Type de résultat
DomResult
SAXResult
StAXResult
StreamResult

- Le choix de type de la source et celui du résultat sont indépendants
- On jour sur la source et sur le résultat pour travailler en mémoire vive ou sur fichier
- Un transformer par défaut fait de la recopie
- Dans ce qui suit, on montre deux transformations :
  - Une sauvegarde d'un objet DOM dans un fichier XML
  - Deux versions de transformation XSLT : v1.0 et v2.0

JAXP : Sauvegarde d'un objet DOM


```
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.*;
public void SauverArbre(String fileName) {
try {
        TransformerFactory tf = TransformerFactory.newInstance();  // fabrique de transformateur
        Transformer transformer = tf.newTransformer();  // creation d'un transformateur
        // Réglages du transformateur
        transformer.setOutputProperty(OutputKeys.METHOD, "xml");
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
        // Sauvegarde de l'objet Document
        File f = new File(???); // ??? Chemin du fichier
        if (f.exists()) { f.delete(); }
        StreamResult nvFichier = new StreamResult(f);
        transformer.transform(new DOMSource(d), nvFichier);
        }
catch(Exception e) { }
```

JAXP : Transformation XSLT


```
import javax.xml.transform.dom.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.*;

        Document dom; // source en mémoire
        String resultFilename; // résultat sur fichier
        String xslFilename; // feuille XSL sur fichier
        DOMSource source = new DOMSource(dom);
         StreamResult result = new StreamResult(new File(resultFilename));
        TransformerFactory tf = TransformerFactory.newInstance();     // XSLT de JAXP (v1.0) (*)
         Transformer transformer = tf.newTransformer(new StreamSource(new File(xslFilename)));
        transformer.transform(source, result);
```

(*) Pour faire une transformation XSLT 2.0, il faut :

- mettre saxon9he.jar dans votre projet (dans le classpath)
- TransformerFactory tf = new net.sf.saxon.TransformerFactoryImpl(); // XSLT de JAXP (v2.0)