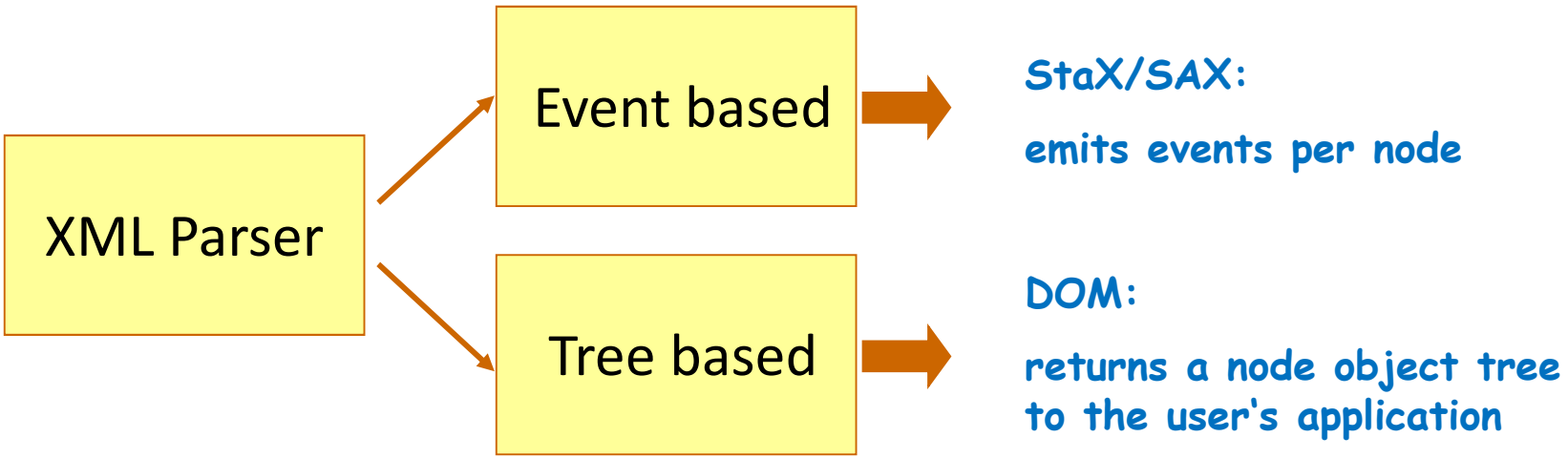
XML – Part B

Processing and Storing

Part 4 – JAXP (Java API for XML Processing)

XML Parser

There are 2 different implementations of XML parsers:



DOM

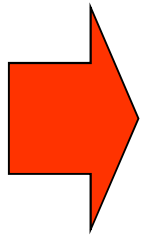
Document Object Model

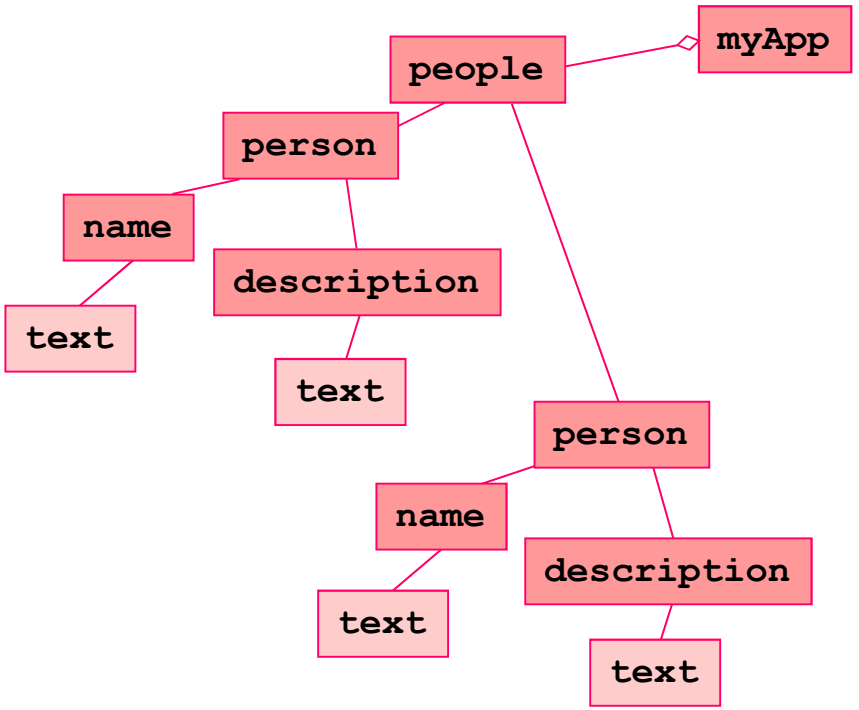
XML Object Models

Objective

- Read an XML document
  - and create corresponding objects
  - for direct use in memory by your program

```
<people>
 <person>
  <name>Winston Churchill</name>
  <description>Winston Churchill
   was a mid 20th century
   politician.
  </description>
 </person>
 <person>
 <name>John F. Kennedy</name>
  <description>JFK was
   a United States president.
  </description>
 </person>
</people>
```





DOM – XML processing (1/2)

Create a new DocumentBuilderFactory

```
// create the DOM factory
final DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

factory.setIgnoringElementContentWhitespace(true);
factory.setNamespaceAware(true);

// Create the schema factory
final SchemaFactory sf =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);

factory.setValidating(true);

// Attach the schema
final Schema schema = sf.newSchema(new File(xsdFileName));
factory.setSchema(schema);
```

Create a SchemaFactory

Activate the schema validation

Attach the schema

DOM – XML processing (2/2)

Create a new DocumentBuilder

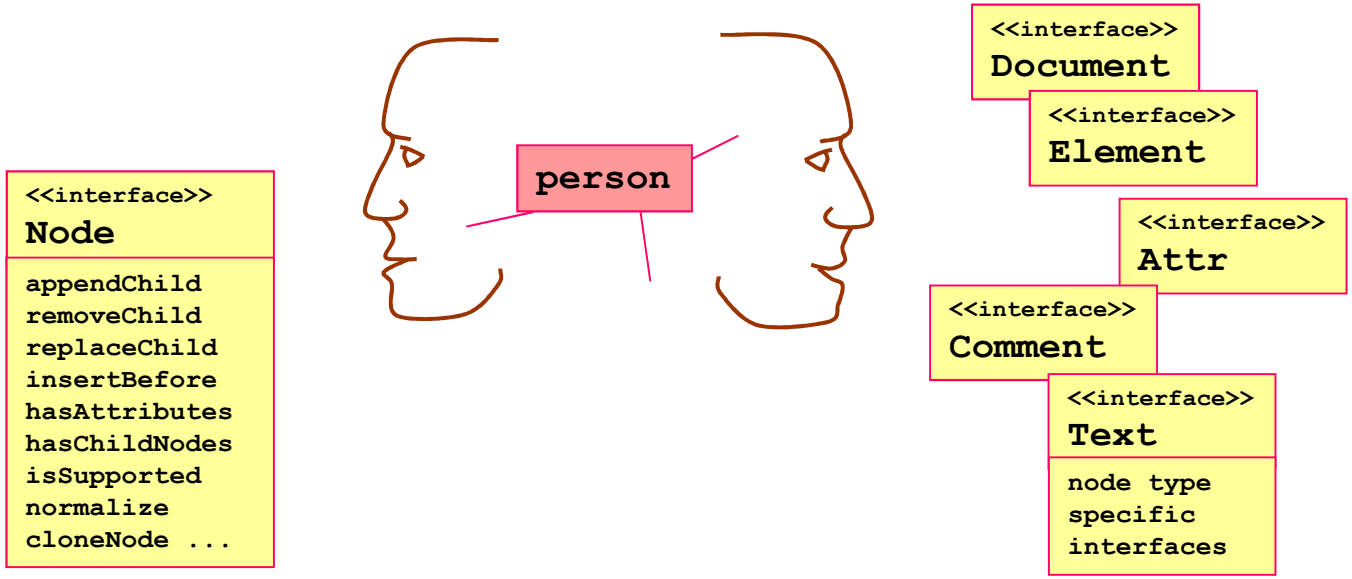
```
// Create the document builder and set the error handler
DocumentBuilder builder = factory.newDocumentBuilder();
builder.setErrorHandler(new DefaultHandler());

// read the xml file
Document doc = builder.parse(new File(xmlFileName));

// process it
// ...
```

Read the XML file

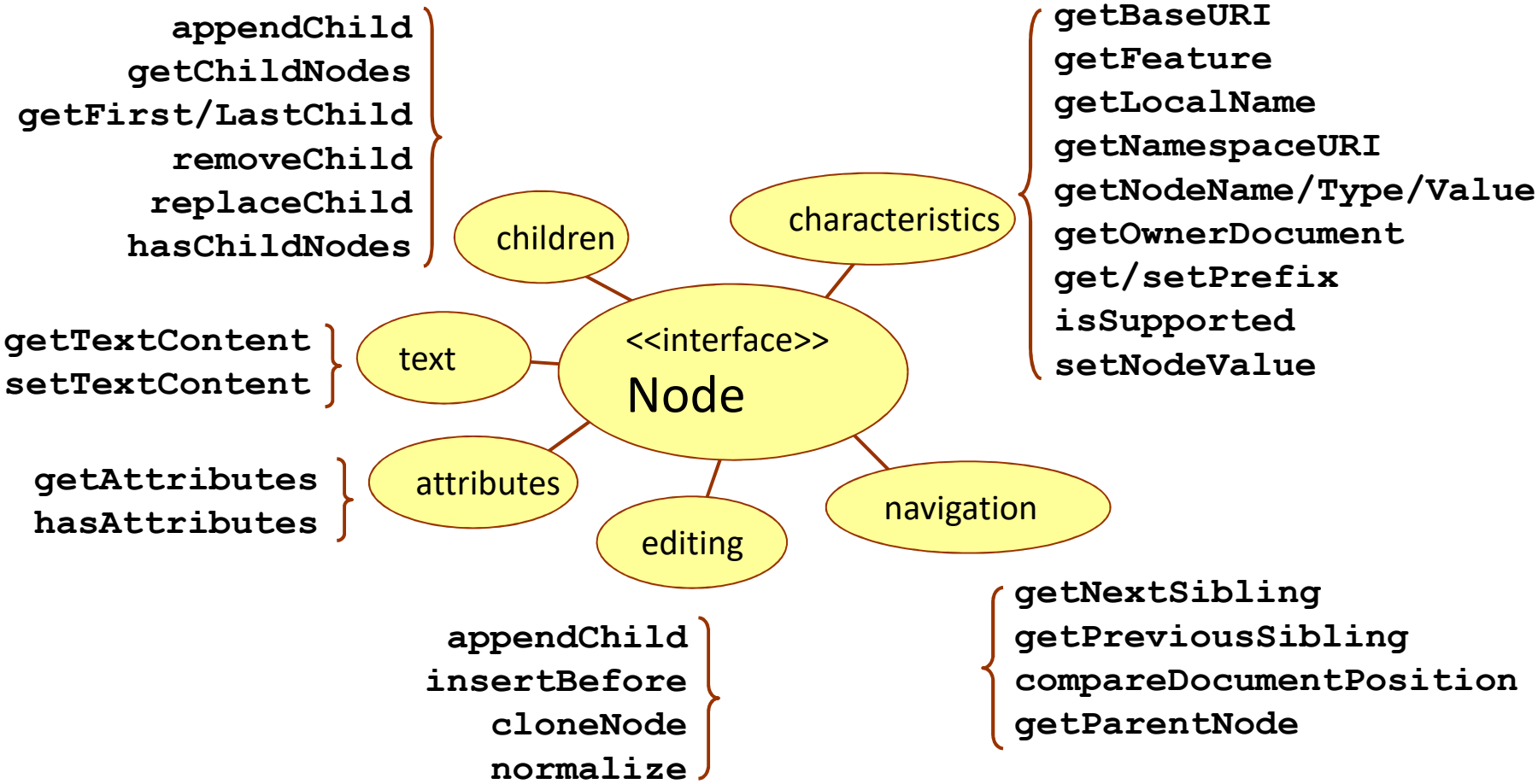
Proces the XML

What is a DOM Object?

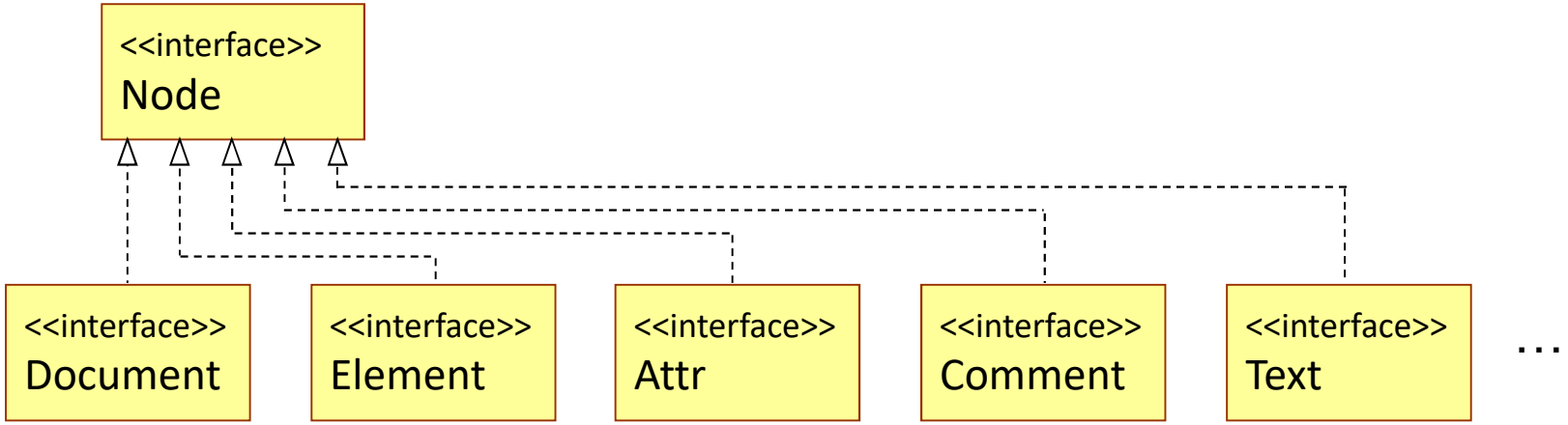
- Each DOM object has two faces
  - it may be considered as a Node
  - or it may be seen as an instantiation of a text, an attribute etc.



The `org.w3c.dom.Node` Interface



The DOM specific Interfaces



`create...` `(all kinds of children)` `get/set data from header etc.`	`get/set attributes`	`getName/ Value` `setValue` `getOwner`	—	`getWholeText` `replaceText` `splitText` `isElementContentWhitespace`

DOM – Node processing

```
// read the xml file
Document doc = builder.parse(new File(xmlFileName));

Node node = doc;

int type = node.getNodeType();

String name = node.getNodeName();

NodeList children = node.getChildNodes();

NamedNodeMap attrs = node.getAttributes();

// Convert the node to a more specific node interface
Element elem = (Element) node;
```

Read the XML file

Convert to a Node

Get the Node type

Get the name

Get the children

Get the attributes

Convert

DOM – Create new DOM

```
// Create document factory and document builder …
// Create document
Document document = documentBuilder.newDocument();

// Create root element with namespace and add it to document
Element personsElement =
    document.createElementNS(PERSON_NS_URI, "pd:Persons");
document.appendChild(personsElement);

// Set schema-related attributes
personsElement.setAttributeNS(XMLConstants.XMLNS_ATTRIBUTE_NS_URI,
    "xmlns:xsi", XMLConstants.W3C_XML_SCHEMA_INSTANCE_NS_URI);
personsElement.setAttributeNS(XMLConstants.W3C_XML_SCHEMA_INSTANCE_NS_URI,
    "xsi:schemaLocation", PERSON_NS_URI + " Schema/Persons.xsd");

// Add further nodes …
```

DOM – Serialize DOM to XML

```
// Get DOM implementation from document
DOMImplementationLS domImplementation =
    (DOMImplementationLS) document.getImplementation();

// Create serializer and configure it
LSSerializer serializer = domImplementation.createLSSerializer();
serializer.getDomConfig().setParameter(
    "format-pretty-print", Boolean.TRUE);

// Create output and set file to write to
LSOutput output = domImplementation.createLSOutput();
output.setByteStream(new FileOutputStream("path to file"));

// Write document to output
serializer.write(document, output);
```

XML Object Models

- There is one official set of - platform- and language-neutral interface - interfaces:

  W3C DOM, currently Level 3
  (s. http://www.w3.org/TR/DOM-Level-3-Core/),
  e.g. in java **org.w3c.dom.***

  - … and a number of implementations:

    Xerces-2, JDOM, dom4j, .NET DOM,
    DomPHP, PyDOM (Python) …

DOM Characteristics

DOM representations of real documents tend to be large

- ➔ they consume much memory space
- ➔ they even may consume too much memory space and then lead to excessive memory page swapping

If possible, read only the necessary parts of your doc

- ➔ StAX (or SAX) + proprietary „mini-model“

Or use JAXB which has a small(er) memory footprint (since it does not store your data separately, and also does a lot of book keeping off line during a compilation step)