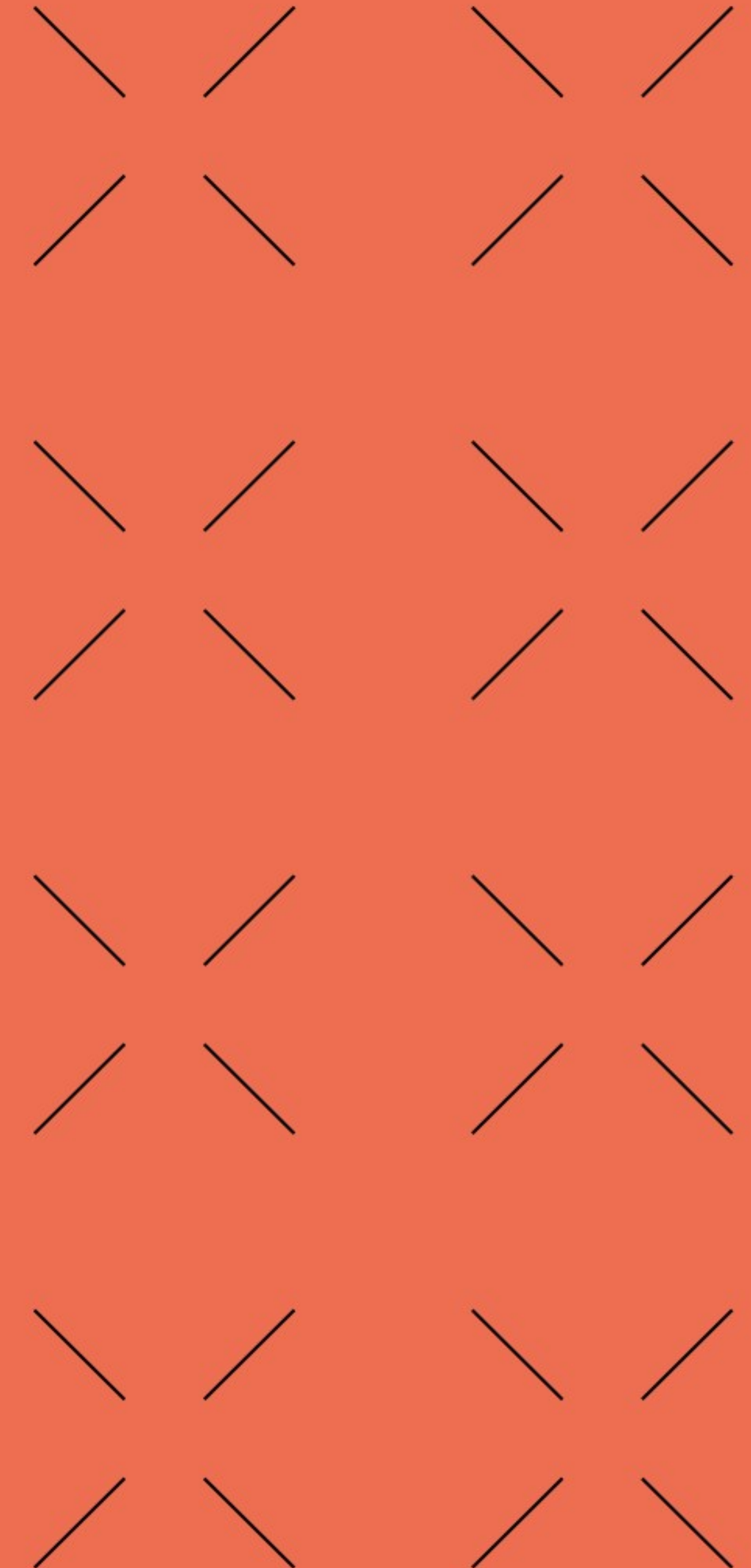


Unit 1. ACCESS TO FILES

Part 2. Files. XML & XSL

Acceso a Datos (ADA) (a distancia en inglés)
CFGS Desarrollo de Aplicaciones Multiplataforma (DAM)

Abelardo Martínez
Year 2023-2024

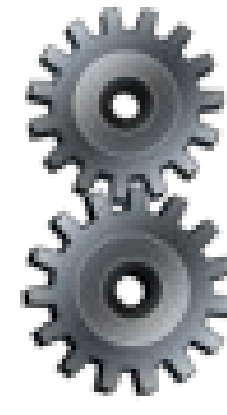
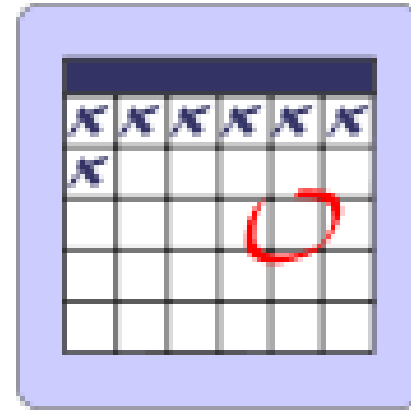


Credits



- Notes made by Abelardo Martínez.
- Based and modified from Sergio Badal (www.sergiobadal.com).
- The images and icons used are protected by the [LGPL](#) licence and have been obtained from:
 - https://commons.wikimedia.org/wiki/Crystal_Clear
 - <https://www.openclipart.org>

Unit progress



FROM	UNIT	WEEKNO.	DESCRIPTION	ASSESSABLE TASKS
	UNIT 1: ACCESS TO FILES			
18/09/23	UNIT 1	WEEK 1	Introduction, Java review, IDE installation and basics of Java	
25/09/23	UNIT 1	WEEK 2	Files and folders. Class File: methods, exceptions. File types	
02/10/23	UNIT 1	WEEK 3	Access types. Reading and writing operations	
09/10/23	UNIT 1	WEEK 4	Files: XML/XSL	
16/10/23	UNIT 1	WEEK 5	Files: XML/XSL	AT1.PRESENTATION
	UNIT 2: ACCESS TO DATABASES			
23/10/23	UNIT 2	WEEK 1	ACCESS TO RELATIONAL DBS	AT2.PRESENTATION
30/10/23	UNIT 2	WEEK 2	ACCESS TO NON RELATIONAL DBS	AT1.SUBMISSION
06/11/23	UNIT 2	WEEK 3	UNIT 1 AND UNIT 2 REVIEW	AT2.SUBMISSION
13/11/23	CONTENTS REVIEW			



Contents

- 1.WHAT IS XML?
- 2.WORKING WITH XML FILES
- 3.SAX. READ XML FILES WITH JAVA
- 4.DOM. MANAGE XML FILES WITH JAVA
 1. READ XML FILES WITH DOM
 2. WRITE XML FILES WITH DOM
 3. MODIFY XML STRUCTURE WITH DOM
- 5.WHAT IS XSL?
- 6.USING XSL TO TRANSFORM DOCUMENTS
- 7.PROPOSED ACTIVITIES
- 8.BIBLIOGRAPHY



1. WHAT IS XML?



XML language

XML is a software and hardware independent tool/language for storing and transporting data

Main features:

- 1) XML stands for **eXtensible Markup Language**
- 2) XML is a **markup language** much like HTML
- 3) XML was **designed to store and transport data** (such JSON or CSV does)
- 4) XML was **designed to be self-descriptive** (intuitive)
- 5) XML is a **W3C recommendation (standard)**
- 6) XML is a **restricted form of SGML**, the Standard Generalized Markup Language [ISO 8879]

More information about XML:

<https://www.w3schools.com/xml/>

The Difference Between XML and HTML

- XML and HTML were designed with different goals:
- **XML was designed to carry data** - with focus on what data is.
 - HTML was designed to display data - with focus on how data looks.
 - XML tags are not predefined like HTML tags are.

XML

```
<firstName>Maria</firstName>
<lastName>Roberts</lastName>
<dateBirth>12-11-1942</dateBirth>
```

HTML

```
<font size="3">Maria Roberts</font>
<b>12-11-1942</b>
```

HTML	VERSUS	XML
HTML		XML
A standard markup language for creating web pages and web applications		A markup language that defines a set of rules for encoding documents in a format that is both human and machine readable
Stands for Hypertext Markup Language		Stands for Extensible Markup Language
Case insensitive		Case sensitive
Has predefined tags		Programmer defines his own set of tag
Some tags do not have a closing tag		It is mandatory to close each tag that has been used
Focuses on displaying data		Focuses on carrying information
Helps to develop the structure of web pages		Helps to exchange data between different platforms
		Visit www.PEDIAA.com

The Difference Between XML and JSON

XML and JSON are two similar ways of storing information.

More information about JSON:

https://www.w3schools.com/whatis/whatis_json.asp

XML	JSON
<pre><empinfo> <employees> <employee> <name>James Kirk</name> <age>40</age> </employee> <employee> <name>Jean-Luc Picard</name> <age>45</age> </employee> <employee> <name>Wesley Crusher</name> <age>27</age> </employee> </employees> </empinfo></pre>	<pre>{ "empinfo" : { "employees" : [{ "name" : "James Kirk", "age" : 40, }, { "name" : "Jean-Luc Picard", "age" : 45, }, { "name" : "Wesley Crusher", "age" : 27, }] } }</pre>

JSON	VERSUS	XML
JSON stands for JavaScript Object Notation and is based on JavaScript language.		XML is short for Extensive Markup Language and is derived from SGML.
It supports text and number data types including integer and strings, and arrays and objects.		It has no direct support for array.
It's a language-independent data-interchange format which supports only UTF-8 encoding.		It's an independent data format which supports different encodings.
It does not contain start and end tags.		It contains start and end tags.
It does not support native objects.		It gets support of objects via attributes and elements.
No support for Namespaces.		Namespaces are supported in XML.
		

2. WORKING WITH XML FILES

Working with XML files

XML files can be used to:

- Provide data to a DB
- Store information in special databases.
- Configuration files
- Exchange of information in web environments (SOAP).
- Execution of commands on remote servers.
- Etc.

To carry out any of these operations, a programming language that provides XML with functionality is necessary (XML is not a complete Turing language).

To access XML files, read their content or alter their structure, an **XML processor** or **XML parser** is used. There are two main ways to work with XML files using Java:

- **SAX**
- **DOM**

XML parsers

1) SAX (Simple API for XML parsing)

A SAX parser (analyser) is an event-based routine to analyse the XML using callbacks. SAX is able to isolate XML data in a single sequential read by detecting the opening and closing hashtags.

Advantages:

- It's very fast.
- **It DOES NOT load the whole document into memory.**
- The best choice for reading large documents.

Disadvantages:

- It has to read the whole document at each query.

2) DOM (Document Object Model)

DOM stores all the data of the XML document in memory in a hierarchical tree structure.

Advantages:

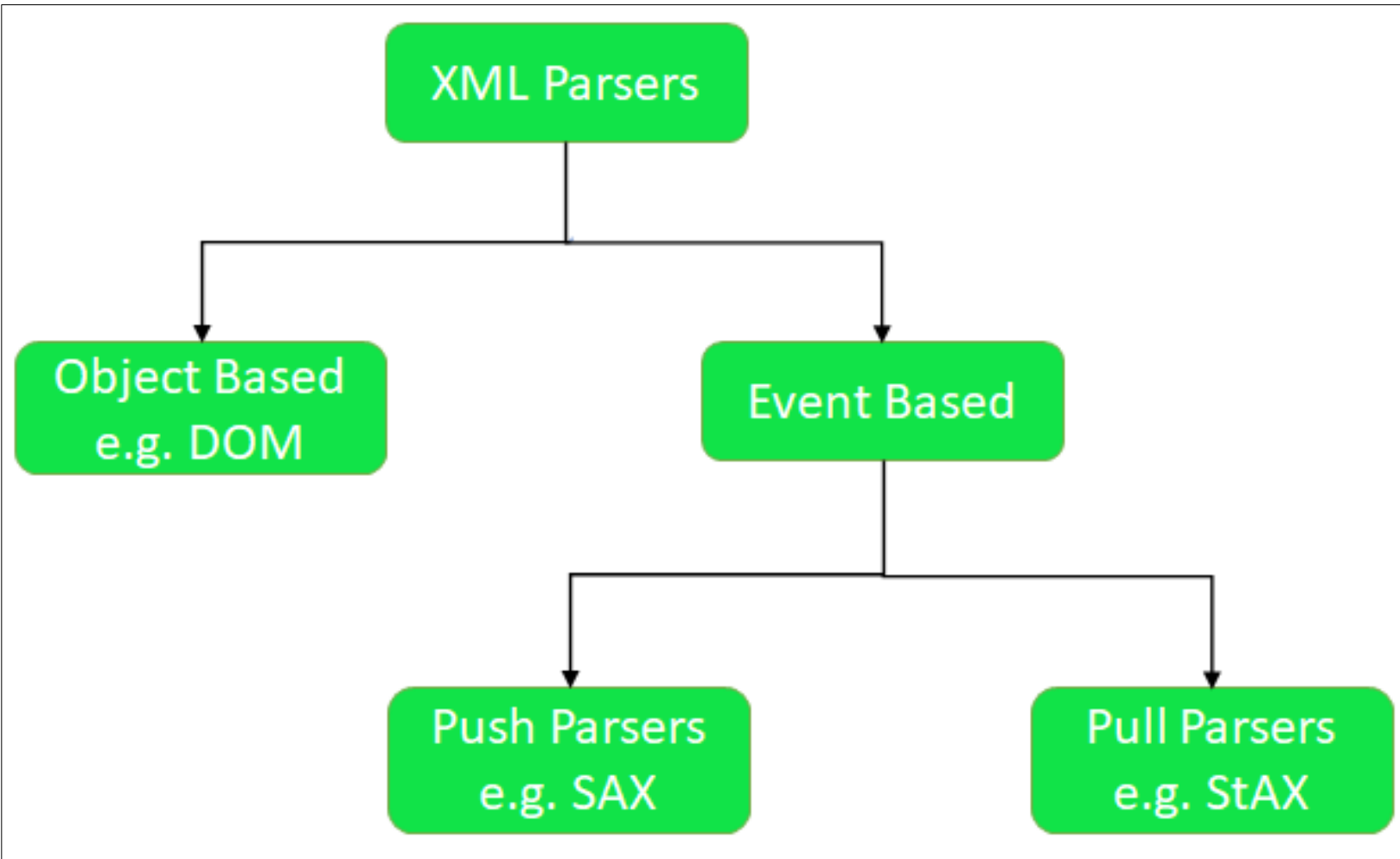
- Simply put, a DOM parser works on the entire XML document.
- It's ideal for applications requiring continuous querying of the data.

Disadvantages:

- **It loads the whole document into memory.**
- Not suitable for large documents.

SAX versus DOM

More information about XML and Java:
<https://www.baeldung.com/java-xml>



DOM Parser	SAX Parser
It reads the entire XML file and creates a tree structure in memory. Therefore, RAM usage is high.	It parses the XML file as a stream and does not allocate space for the entire file in memory.
It runs slow as the entire file is placed in memory and is not an efficient method for large XML files.	Since the entire file is not placed in memory, it requires less memory and is faster than the DOM Parser.
Some versions have methods that can update the XML file.	SAX Parser does not have update methods.
Since it can access the entire file, elements can be accessed randomly.	Since the entire file is not stored in memory, sequential access to elements is possible.

	SAX	DOM
Memory efficient	ok	no
Bidirectional navigation	no	ok
xml manipulation	no	ok
data binding	no	no

3. SAX. READ XML FILES WITH JAVA

File access with SAX

The SAX API is fully included within the JRE.

- If we give the XML file to the API **it will not do anything**.
- On load, some methods will be called for every item found.
- All these methods are defined in the **DefaultHandler**.

Credits & code (How to read XML file in Java (SAX Parser):

<https://mkyong.com/java/how-to-read-xml-file-in-java-sax-parser/>

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
  <staff id="1001">
    <name>mkyong</name>
    <role>support</role>
    <salary currency="USD">5000</salary>
    <!-- for special characters like < &, need CDATA -->
    <bio><![CDATA[HTML tag <code>testing</code>]]></bio>
  </staff>
  <staff id="1002">
    <name>yflow</name>
    <role>admin</role>
    <salary currency="EUR">8000</salary>
    <bio><![CDATA[a & b]]></bio>
  </staff>
</Company>
```

```
public class ReadXmlSaxParser {

    private static final String FILENAME = "src/main/resources/staff.xml";

    public static void main(String[] args) {

        SAXParserFactory factory = SAXParserFactory.newInstance();

        try {

            SAXParser saxParser = factory.newSAXParser();

            PrintAllHandlerSax handler = new DefaultHandler ();
            saxParser.parse(FILENAME, handler);

        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }

    }

}
```

File access with SAX

Steps to read an XML file with SAX:

- The first thing we will do is to import all the necessary classes and interfaces.
- We will create a class to extend the **DefaultHandler**.
- Then, we will override the **startDocument**, **endDocument**, **startElement**, **endElement** and **characters** methods.
- With this, we will be able to print all the XML elements, attributes, comments and texts.

```
import org.xml.sax.SAXException;
import javax.xml.parsers.SAXParser;
```

Java version:  JRE System Library [JavaSE-1.8]

```
public class PrintAllHandlerSax extends DefaultHandler {

    private StringBuilder currentValue = new StringBuilder();

    @Override
    public void startDocument() {
        System.out.println("Start Document");
    }

    @Override
    public void endDocument() {
        System.out.println("End Document");
    }
}
```

```
public class ReadXmlSaxParser {

    private static final String FILENAME = "src/main/resources/staff.xml";

    public static void main(String[] args) {

        SAXParserFactory factory = SAXParserFactory.newInstance();

        try {

            SAXParser saxParser = factory.newSAXParser();

            PrintAllHandlerSax handler = new PrintAllHandlerSax();
            saxParser.parse(FILENAME, handler);

        } catch (ParserConfigurationException | SAXException | IOException e) {
            e.printStackTrace();
        }

    }
}
```

File access with SAX

Output:

- This could be the result placing a simple **println** on every method but we could store the information on a database, save it on another device or send it elsewhere.

```
<?xml version="1.0" encoding="utf-8"?>
<Company>
  <staff id="1001">
    <name>mkyong</name>
    <role>support</role>
    <salary currency="USD">5000</salary>
    <!-- for special characters like < &, need CDATA -->
    <bio><![CDATA[HTML tag <code>testing</code>]]></bio>
  </staff>
  <staff id="1002">
    <name>yflow</name>
    <role>admin</role>
    <salary currency="EUR">8000</salary>
    <bio><![CDATA[a & b]]></bio>
  </staff>
</Company>
```



```
Start Document
Start Element : Company
Start Element : staff
Staff id : 1001
Start Element : name
End Element : name
Name : mkyong
Start Element : role
End Element : role
Role : support
Start Element : salary
Currency :USD
End Element : salary
Salary : 5000
Start Element : bio
End Element : bio
Bio : HTML tag <code>testing</code>
End Element : staff
Start Element : staff
Staff id : 1002
Start Element : name
End Element : name
Name : yflow
Start Element : role
End Element : role
Role : admin
Start Element : salary
Currency :EUR
End Element : salary
Salary : 8000
Start Element : bio
End Element : bio
Bio : a & b
End Element : staff
End Element : Company
End Document
```

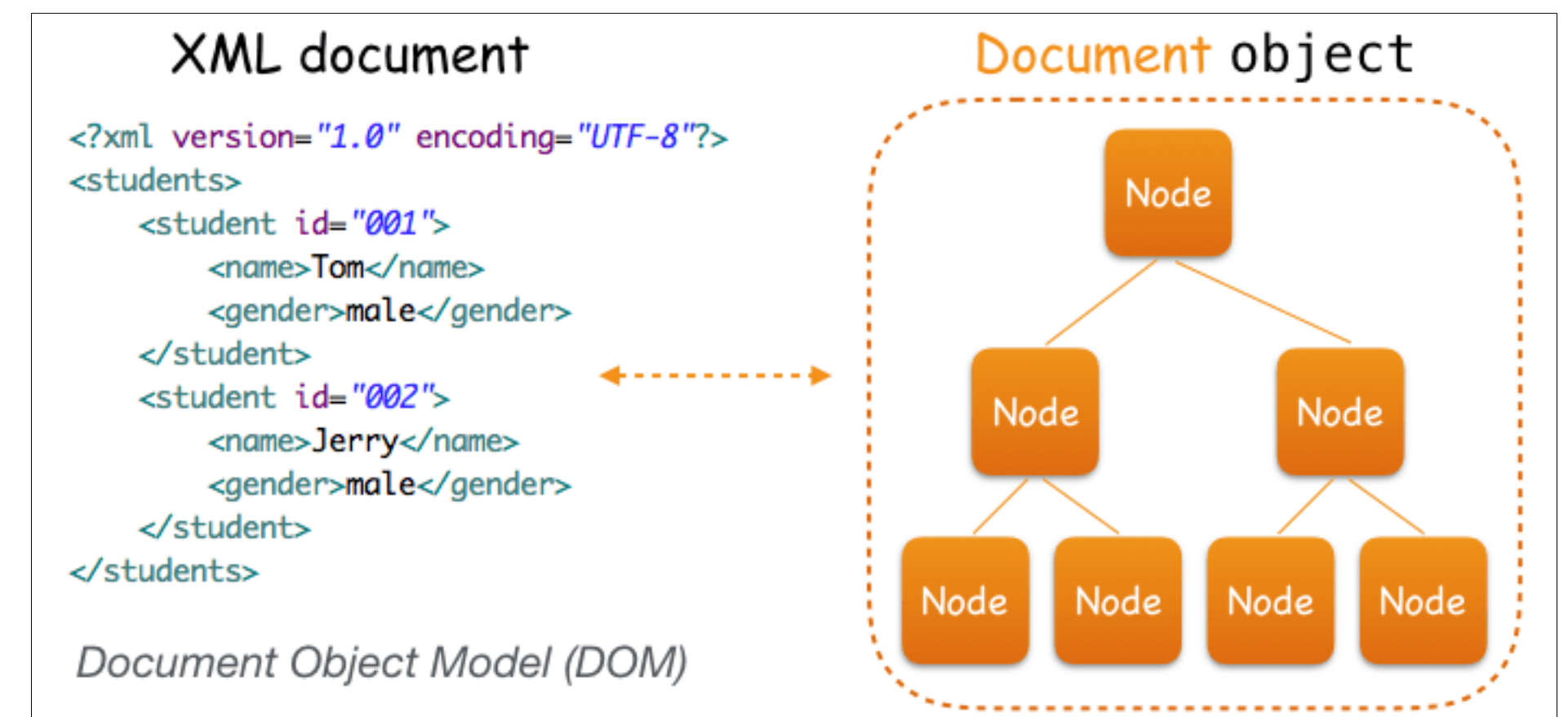
4. DOM. MANAGE XML FILES WITH JAVA

File access with DOM

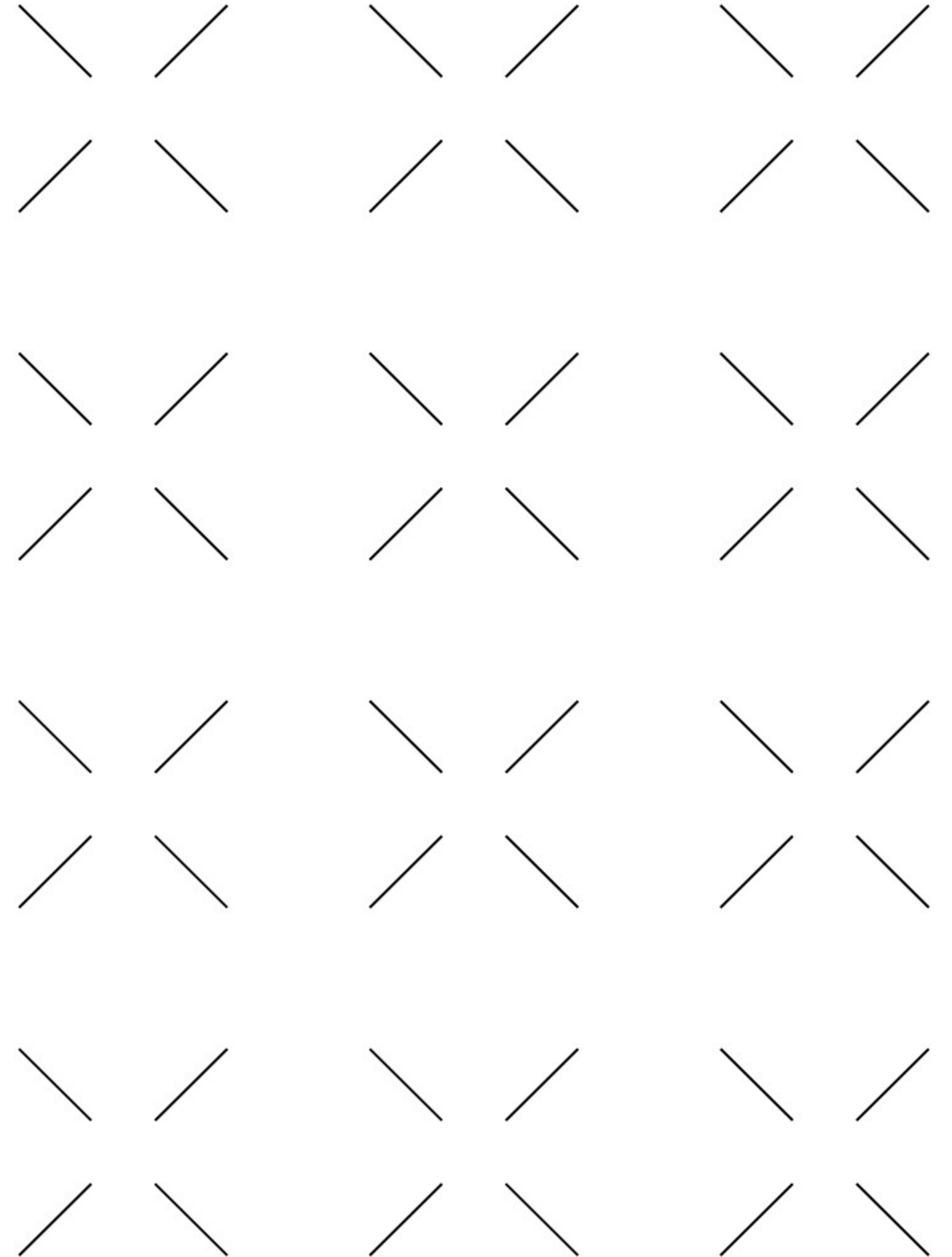
- DOM parser is intended for working with XML as an object.
- This object is a graph (a tree like structure) in memory so called “Document Object Model (DOM)”.
- The procedure is quite similar to the one used in JavaScript:
 - 1) First, the parser goes through the whole XML file creating DOM objects corresponding to the nodes in the XML file.
 - These DOM objects are linked together in a tree like structure.
 - 2) Once the parser is done with parsing process we get this tree-like DOM object structure.
 - Now we can “walk” the DOM structure back and forth as we want to get/update/delete data from it.

Credits & code (Java DOM Parser Example):

<https://howtodoinjava.com/java/xml/read-xml-dom-parser-example/>



4.1 Read XML files with DOM



File reading with DOM

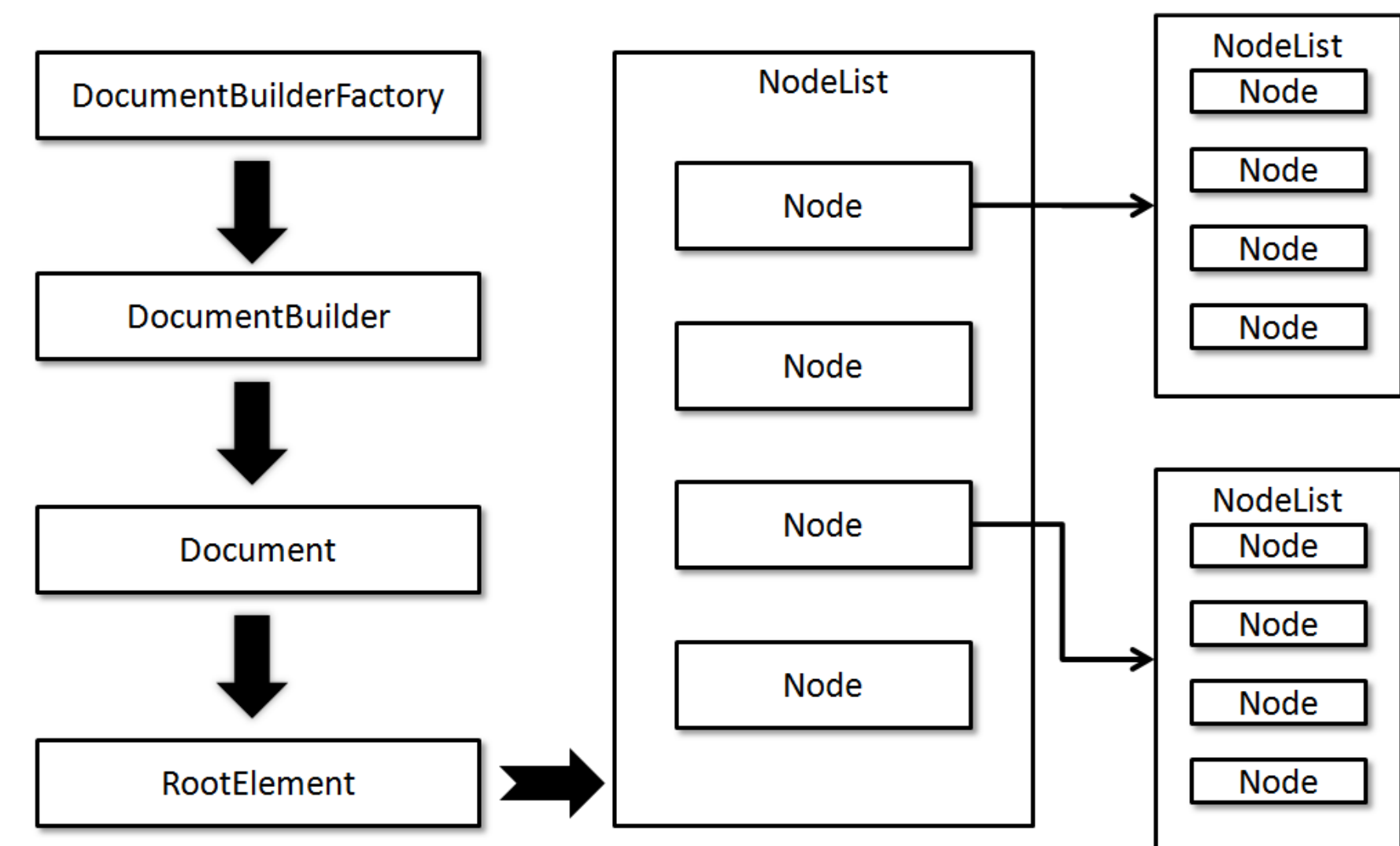
Steps to read an XML file with DOM:

- Create DocumentFactory (load XML to memory)
- Create DocumentBuilder
- Create Document object
- Extract root element
- Walk through the document
 - ➔ Examine attributes
 - ➔ Examine sub-elements
 - ➔ Repeat recursively

Credits & code (Java DOM Parser Example):

<https://howtodoinjava.com/java/xml/read-xml-dom-parser-example/>

(Document Object Model – DOM) XML Element Parsing Process



File reading with DOM

Full example:

```
import org.w3c.dom.Document;  
import org.w3c.dom.Element;
```

Java version:  JRE System Library [JavaSE-1.8]

```
<employees>  
  <employee id="111">  
    <firstName>Lokesh</firstName>  
    <lastName>Gupta</lastName>  
    <location>India</location>  
  </employee>  
  <employee id="222">  
    <firstName>Alex</firstName>  
    <lastName>Gussin</lastName>  
    <location>Russia</location>  
  </employee>  
  <employee id="333">  
    <firstName>David</firstName>  
    <lastName>Feezor</lastName>  
    <location>USA</location>  
  </employee>  
</employees>
```



```
//Get Document Builder  
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  
DocumentBuilder builder = factory.newDocumentBuilder();  
  
//Build Document  
Document document = builder.parse(new File("employees.xml"));
```

```
//Here comes the root node  
Element root = document.getDocumentElement();  
System.out.println(root.getNodeName());  
  
//Get all employees  
NodeList nList = document.getElementsByTagName("employee");  
System.out.println("=====");
```

```
for (int temp = 0; temp < nList.getLength(); temp++)  
{  
  Node node = nList.item(temp);  
  System.out.println("");    //Just a separator  
  if (node.getNodeType() == Node.ELEMENT_NODE)  
  {  
    //Print each employee's detail  
    Element eElement = (Element) node;  
    System.out.println("Employee id : " + eElement.getAttribute("id"));  
    System.out.println("First Name : " + eElement.getElementsByTagName("firstName").item(0).getTextContent());  
    System.out.println("Last Name : " + eElement.getElementsByTagName("lastName").item(0).getTextContent());  
    System.out.println("Location : " + eElement.getElementsByTagName("location").item(0).getTextContent());  
  }  
}
```

File reading with DOM

Output:

- This could be the result placing a simple **println** on every element but we could store the information on a database, save it on another device or send it elsewhere.

```
<employees>
  <employee id="111">
    <firstName>Lokesh</firstName>
    <lastName>Gupta</lastName>
    <location>India</location>
  </employee>
  <employee id="222">
    <firstName>Alex</firstName>
    <lastName>Gussin</lastName>
    <location>Russia</location>
  </employee>
  <employee id="333">
    <firstName>David</firstName>
    <lastName>Feezor</lastName>
    <location>USA</location>
  </employee>
</employees>
```



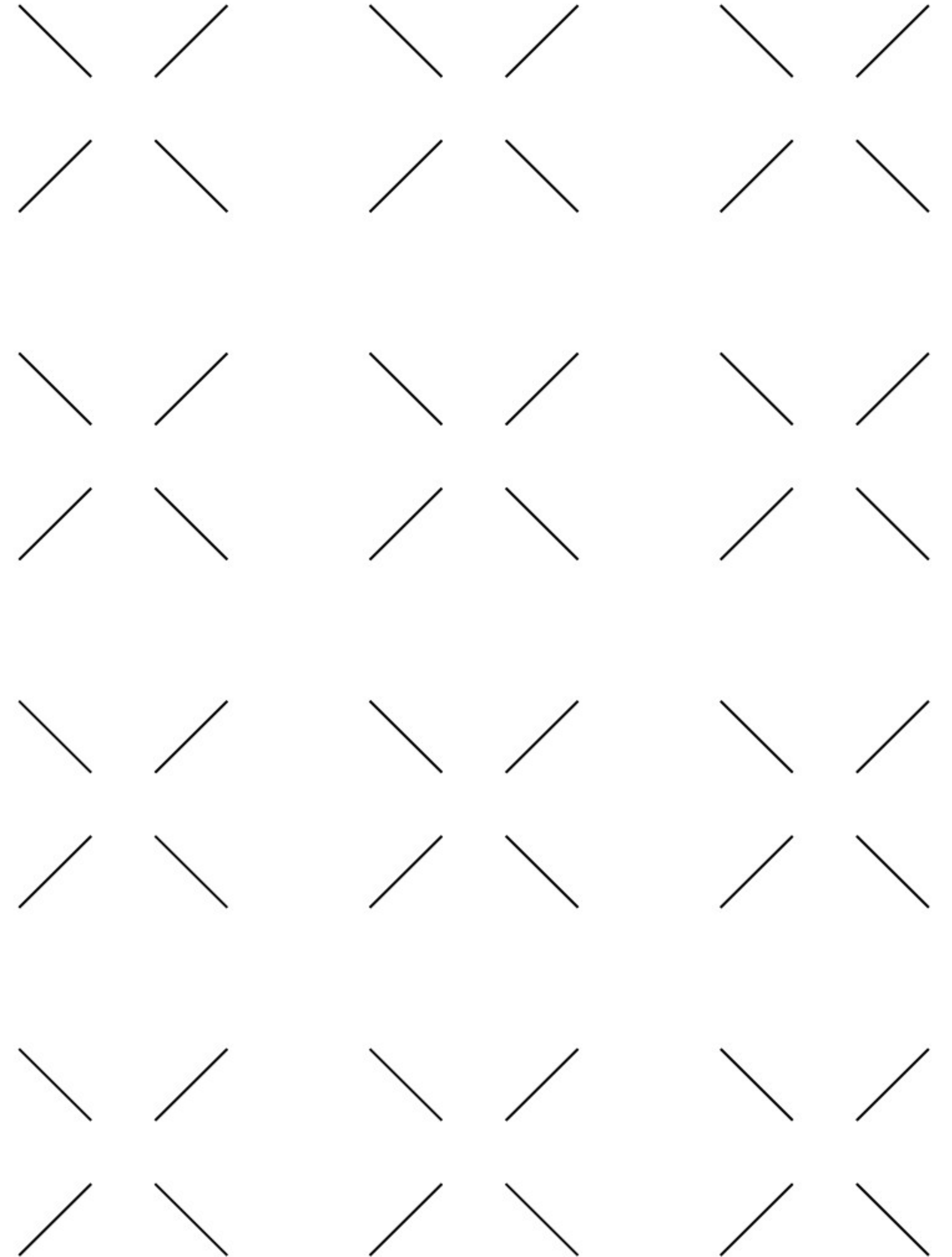
```
employees
=====

Employee id : 111
First Name : Lokesh
Last Name : Gupta
Location : India

Employee id : 222
First Name : Alex
Last Name : Gussin
Location : Russia

Employee id : 333
First Name : David
Last Name : Feezor
Location : USA
```

4.2 Write XML files with DOM



File writing with DOM

Steps to create and write XML to a file:

- Create a Document doc (load XML to memory).
- Create XML elements, attributes, etc., and append to the Document doc.
- Create a Transformer to write the Document doc to disk.

Credits & code (Create XML file in java DOM):

<https://mkyong.com/java/how-to-create-xml-file-in-java-dom/>

1) Creation (load XML to memory):

```
public static void main(String[] args)
    throws ParserConfigurationException, TransformerException {

    DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

    // root elements
    Document doc = docBuilder.newDocument();
    Element rootElement = doc.createElement("company");
    doc.appendChild(rootElement);

    doc.createElement("staff");
    rootElement.appendChild(doc.createElement("staff"));

    //...create XML elements, and others...

    // write dom document to a file
    try (FileOutputStream output =
        new FileOutputStream("c:\\test\\staff-dom.xml")) {
        writeXml(doc, output);
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

File writing with DOM

2) Writing:

```
// write doc to output stream
private static void writeXml(Document doc,
                              OutputStream output)
    throws TransformerException {

    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(output);

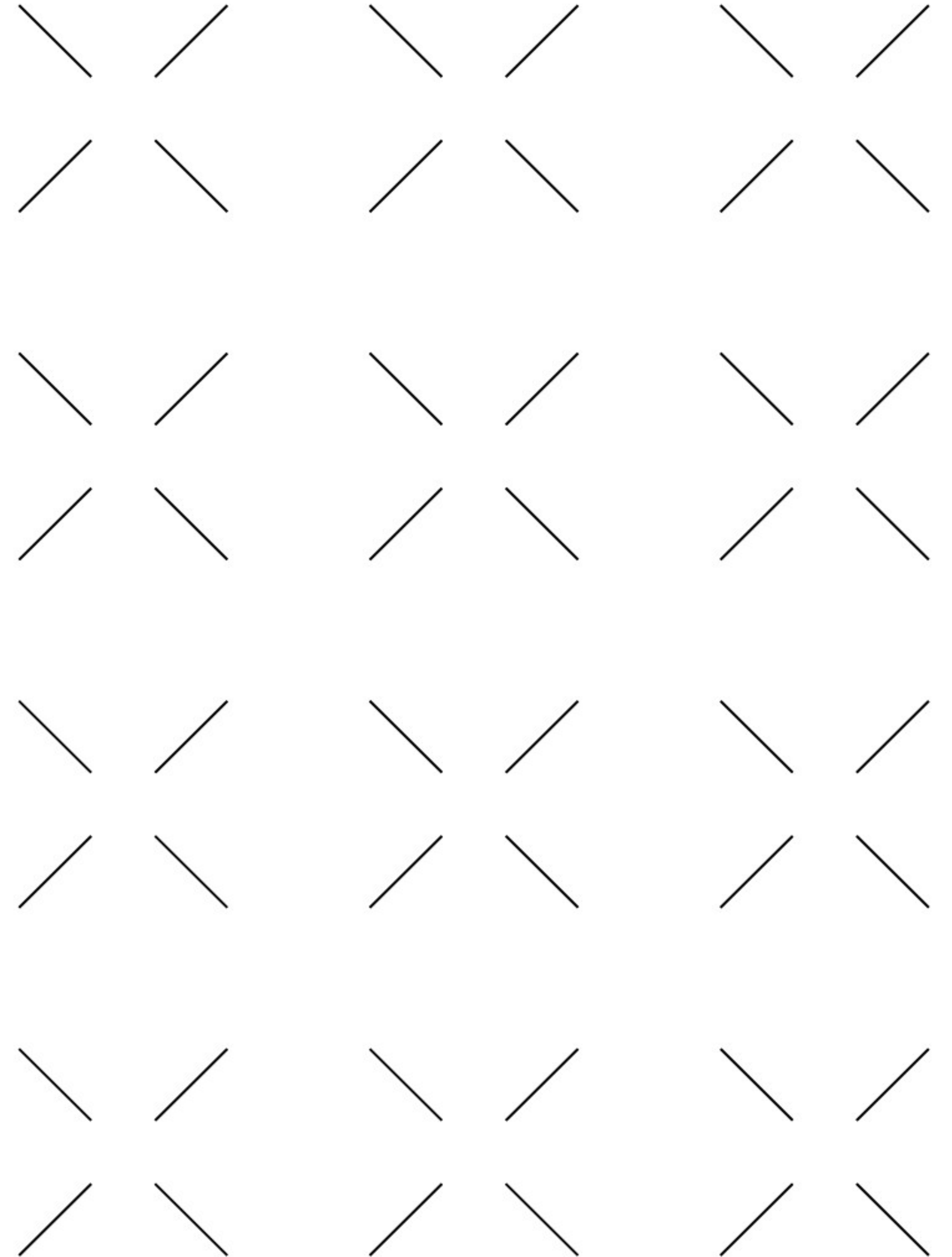
    transformer.transform(source, result);

}
```



```
transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.setOutputProperty(OutputKeys.METHOD, "xml");
transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "5");
```

4.3 Modify XML structure with DOM



File modification with DOM

Steps to change several values only within specific node(s):

- Create a Document doc (load XML to memory).
- Loop for all subnodes.
- If specific node(s) is found, change the values of its subnodes.
- Save XML to disk.

Credits & code (Update nodes and attributes with DOM):

<https://learningprogramming.net/java/dom/update-nodes-and-attributes-with-dom-in-java-xml/>

1) Creation (load XML to memory):

```
<?xml version="1.0" encoding="UTF-8"?>
<products>
  <product>
    <id>p01</id>
    <name>name 1</name>
    <price currency="usd">20</price>
    <quantity>5</quantity>
    <weight unit="kg">2.1</weight>
    <date format="dd/MM/yyyy">02/03/2017</date>
    <status>true</status>
  </product>
  <product>
    <id>p02</id>
    <name>name 2</name>
    <price currency="eur">12</price>
    <quantity>3</quantity>
    <weight unit="kg">6.5</weight>
    <date format="dd/MM/yyyy">24/11/2018</date>
    <status>true</status>
  </product>
</products>
```

File modification with DOM

2) XML structure walkthrough:

```
private static void update(String id, String newCurrency, double newPrice, boolean newStatus) {
    String xmlFile = "src\\data\\products.xml";
    try {
        DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();
        DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();
        Document document = documentBuilder.parse(xmlFile);
        NodeList products = document.getElementsByTagName("product");
        for (int i = 0; i < products.getLength(); i++) {
            Element product = (Element) products.item(i);
            if (product.getElementsByTagName("id").item(0).getTextContent().equalsIgnoreCase(id))

                // Update value of price tag and currency attribute
                Element priceTag = (Element) product.getElementsByTagName("price").item(0);
                priceTag.setTextContent(String.valueOf(newPrice));
                priceTag.setAttribute("currency", newCurrency);

                // Update value of status tag
                Element statusTag = (Element) product.getElementsByTagName("status").item(0);
                statusTag.setTextContent(String.valueOf(newStatus));

                break;
            }
        }

        saveXMLContent(document, xmlFile);
    } catch (Exception e) {
        System.err.println(e.getMessage());
    }
}
```



3) Update changes and save:

```
public static void main(String[] args) {

    update("p02", "euro", 999, false);

}
```

```
private static void saveXMLContent(Document document, String xmlFile) {
    try {
        TransformerFactory transformerFactory = TransformerFactory.newInstance();
        Transformer transformer = transformerFactory.newTransformer();
        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
        DOMSource domSource = new DOMSource(document);
        StreamResult streamResult = new StreamResult(xmlFile);
        transformer.transform(domSource, streamResult);
    } catch (Exception ex) {
        System.out.println(ex.getMessage());
    }
}
```


5. WHAT IS XSL?

XSL language

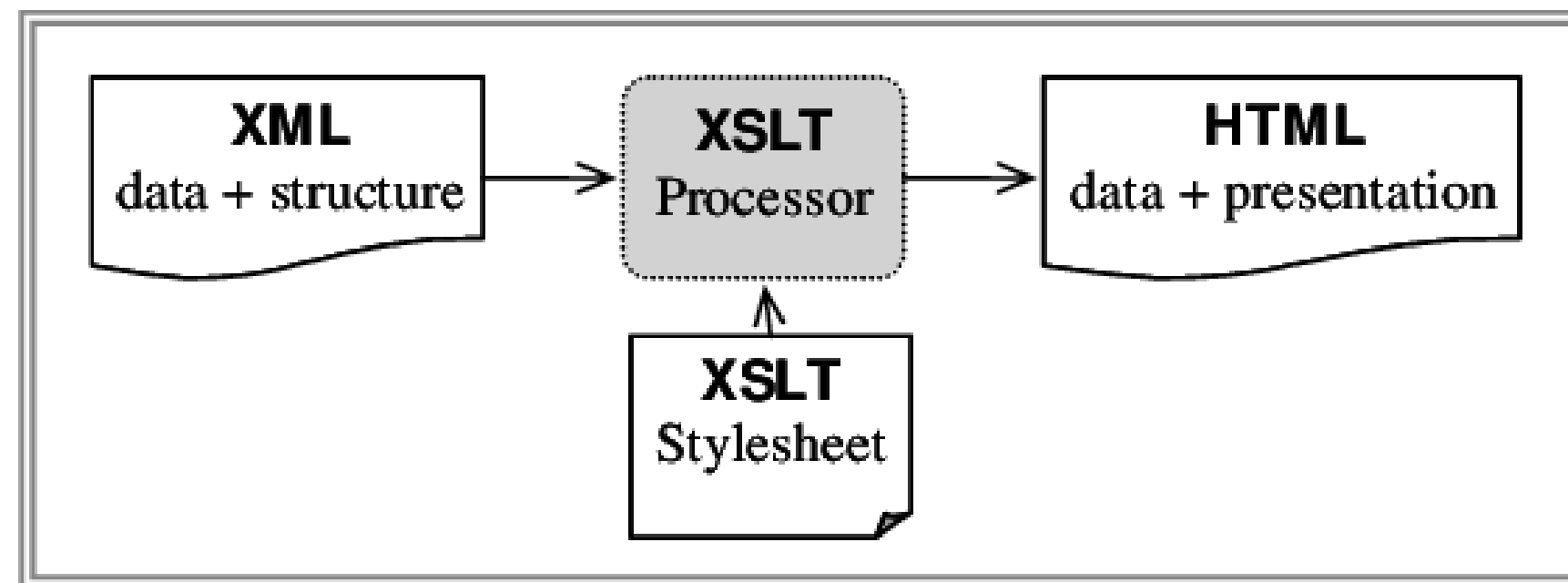
XSL is a styling language for XML:

- XSL stands for eXtensible Stylesheet Language.
- XSL is a **styling language** for XML just like CSS is a styling language for HTML.
- **XSLT** stands for **XSL Transformation** and it is used to transform XML documents into other formats (XML => HTML)

More information about XSL:

https://www.w3schools.com/xml/xsl_intro.asp

<https://www.tutorialspoint.com/xslt/index.htm>



XSL vs. CSS

- Similar problem, different solutions
- CSS takes HTML and applies fonts, styles, positions
- XSL takes any XML and turns it into anything else
- XSL more powerful than CSS
 - e.g. can use same content in multiple places in result document

What is the required level of XSL?

XSL/XSLT can be as complex as CSS.

Learning deeply XSL is NOT the aim of this UNIT.



Our purpose is:

- Understand what is XSL/XSLT doing.
- Learn some basics of XSL/XSLT.
- Learn how to apply an XSL file to an XML file to get an HTML using Java.

We will present the basics of XSL/Java working with this example:

<http://javaonlineguide.net/2016/02/convert-xml-to-html-in-java-using-xslt-example.html>

You can go further on XSL/Java studying this example:

<https://www.oreilly.com/library/view/learning-java-4th/9781449372477/ch24s10.html>

Working with XSL

Let's present an example to understand what is actually doing XSLT.

Credits & code (XSLT Basic Example):

https://developer.mozilla.org/en-US/docs/Web/API/XSLTProcessor/Basic_Example

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="example.xsl"?>
<Article>
  <Title>My Article</Title>
  <Authors>
    <Author>Mr. Foo</Author>
    <Author>Mr. Bar</Author>
  </Authors>
  <Body>This is my article text.</Body>
</Article>
```

example.xml

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="text"/>

  <xsl:template match="/">
    Article - <xsl:value-of select="/Article/Title"/>
    Authors: <xsl:apply-templates select="/Article/Authors/Author"/>
  </xsl:template>

  <xsl:template match="Author">
    - <xsl:value-of select="." />
  </xsl:template>

</xsl:stylesheet>
```

example.xsl



```
Article - My Article
Authors:
- Mr. Foo
- Mr. Bar
```

output

6. USING XSL TO TRANSFORM DOCUMENTS

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

1) First, let's have a look to the XML. As you can see, it's very plain and simple.

Credits & code (Convert XML to HTML in Java using XSLT):

<http://javaonlineguide.net/2016/02/convert-xml-to-html-in-java-using-xslt-example.html>

```
<ProductList>
  <Product>
    <productId>I1</productId>
    <productName>Samsung LED Tv</productName>
    <price>40000.00</price>
    <stock>5</stock>
  </Product>
  <Product>
    <productId>I3</productId>
    <productName>SONY LCD TV</productName>
    <price>30000.00</price>
    <stock>7</stock>
  </Product>
</ProductList>
```

XML

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

2) **Second**, let's study what kind of output (HTML) we want to get.

XML

```
<ProductList>
  <Product>
    <productId>I1</productId>
    <productName>Samsung LED Tv</productName>
    <price>40000.00</price>
    <stock>5</stock>
  </Product>
  <Product>
    <productId>I3</productId>
    <productName>SONY LCD TV</productName>
    <price>30000.00</price>
    <stock>7</stock>
  </Product>
</ProductList>
```



XSL?

Output (HTML)

Product Code:	Product Name:	Price:	Stock:
I1	Samsung LED Tv	40000.00	5
I3	SONY LCD TV	30000.00	7

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

2) **Second**, let's study what kind of output (HTML) we want to get.

HTML

Output (HTML)

Product Code:	Product Name:	Price:	Stock:
I1	Samsung LED Tv	40000.00	5
I3	SONY LCD TV	30000.00	7

And the CSS?



```
<html>
  <body>
    <table class="tfmt">
      <tr>
        <th style="width:250px">Product Code:</th>
        <th style="width:350px">Product Name:</th>
        <th style="width:250px">Price:</th>
        <th style="width:250px">Stock:</th>
      </tr>
      <tr>
        <td class="colfmt">I1</td>
        <td class="colfmt">Samsung LED Tv</td>
        <td class="colfmt">40000.00</td>
        <td class="colfmt">5</td>
      </tr>
      <tr>
        <td class="colfmt">I3</td>
        <td class="colfmt">SONY LCD TV</td>
        <td class="colfmt">30000.00</td>
        <td class="colfmt">7</td>
      </tr>
    </table>
  </body>
</html>
```

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

2) **Second**, let's study what kind of output (HTML) we want to get.

Output (HTML)

Product Code:	Product Name:	Price:	Stock:
I1	Samsung LED Tv	40000.00	5
I3	SONY LCD TV	30000.00	7



HTML + CSS

```
<html>
<head>
  <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <style type="text/css">
    table.tfmt {
      border: 1px ;
    }

    td.colfmt {
      border: 1px ;
      background-color: white;
      color: black;
      text-align:right;
    }

    th {
      background-color: #2E9AFE;
      color: white;
    }
  </style>
</head>
<body>
  <table class="tfmt">
    <tr>
      <th style="width:250px">Product Code:</th>
      <th style="width:350px">Product Name:</th>
      <th style="width:250px">Price:</th>
      <th style="width:250px">Stock:</th>
    </tr>
    <tr>
      <td class="colfmt">I1</td>
      <td class="colfmt">Samsung LED Tv</td>
      <td class="colfmt">40000.00</td>
      <td class="colfmt">5</td>
    </tr>
    <tr>
      <td class="colfmt">I3</td>
      <td class="colfmt">SONY LCD TV</td>
      <td class="colfmt">30000.00</td>
      <td class="colfmt">7</td>
    </tr>
  </table>
</body>
</html>
```

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

3) Third, define the XSL file.

Basically, **xsl:template match="/"** executes at the beginning, adding the headers of the document and the table.

Then, **xsl:for-each** node matching the criteria, a new row (tr) is added with its columns (tr) and data (**xsl:value-of**).

```
<tr>
  <td class="colfmt">
    <xsl:value-of select="productId" />
  </td>
  <td class="colfmt">
    <xsl:value-of select="productName" />
  </td>

  <td class="colfmt">
    <xsl:value-of select="price" />
  </td>
  <td class="colfmt">
    <xsl:value-of select="stock" />
  </td>
</tr>
```



XSL

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <head>
        <style type="text/css">
          table.tfmt {
            border: 1px ;
          }

          td.colfmt {
            border: 1px ;
            background-color: white;
            color: black;
            text-align:right;
          }

          th {
            background-color: #2E9AFE;
            color: white;
          }
        </style>
      </head>
      <body>
        <table class="tfmt">
          <tr>
            <th style="width:250px">Product Code:</th>
            <th style="width:350px">Product Name:</th>
            <th style="width:250px">Price:</th>
            <th style="width:250px">Stock:</th>
          </tr>
          <xsl:for-each select="ProductList/Product">

            </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Convert XML to HTML in Java using XSLT

In this example, we will convert XML to HTML using XSLT language.

4) Fourth, apply the transformations.

To transform the XML into this particular HTML we have to join the XML, with the XSL and let Java do the rest.

We have to create:

- One StreamSource for the XML
- One StreamSource for the XSL
- One StringWriter to hold the transformation
- One FileWriter for the resulting HTML

A TransformerFactory instance will do the rest in a very intuitive way.

Keep in mind all the exceptions you have to manage.

```
public static void main(String args[]) {
    Source xml = new StreamSource(new File("D:\\template\\product.xml"));
    Source xslt = new StreamSource("D:\\template\\product.xsl");

    convertXMLToHTML(xml, xslt);

}

public static void convertXMLToHTML(Source xml, Source xslt) {
    StringWriter sw = new StringWriter();

    try {

        FileWriter fw = new FileWriter("D:\\template\\product.html");
        TransformerFactory tFactory = TransformerFactory.newInstance();
        Transformer transform = tFactory.newTransformer(xslt);
        transform.transform(xml, new StreamResult(sw));
        fw.write(sw.toString());
        fw.close();

        System.out
            .println("product.html generated successfully at D:\\template ");

    } catch (IOException | TransformerConfigurationException e) {
        e.printStackTrace();
    } catch (TransformerFactoryConfigurationError e) {
        e.printStackTrace();
    } catch (TransformerException e) {
        e.printStackTrace();
    }
}
```

7. PROPOSED ACTIVITIES

Proposed activities



Check the suggested exercises you will find at the “Aula Virtual”. **These activities are optional and non-assessable but** understanding these non-assessable activities is essential to solve the assessable task ahead.

Shortly you will find the proposed solutions enclosed in a single PDF.

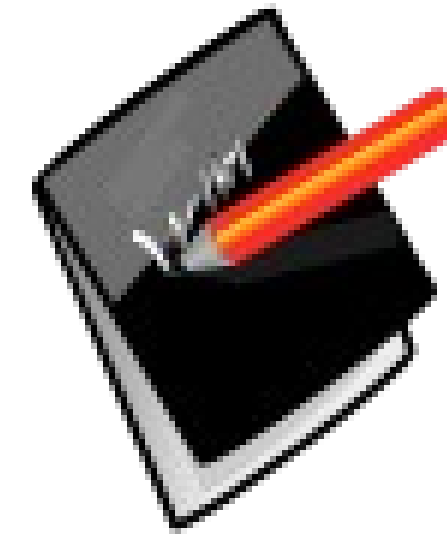
What Now?



This week you should ...

- 1) Check the solutions of the exercises suggested last week
- 2) Study this document and every external resource you need to understand the contents suggested for this week.
- 3) Try to do the suggested exercises. Go to the UNIT forum to share your doubts and alternative implementations with the rest of the students.
- 4) Check the materials for next week BEFORE attending to the next TC.

8. BIBLIOGRAPHY



Resources

- W3 Schools. Introduction to XML language. <https://www.w3schools.com/xml/>
- Abrir llave. Tutorial XML. <https://www.abrirllave.com/>
- Tutorials point. XML – Tutorial. <https://www.tutorialspoint.com/es/xml/index.htm>
- Oracle Documentation. SAX. <https://docs.oracle.com/javase/tutorial/jaxp/sax/index.html>
- Oracle Documentation. DOM. <https://docs.oracle.com/javase/tutorial/jaxp/dom/index.html>
- Josep Cañellas Bornas, Isidre Guixà Miranda. Accés a dades. Desenvolupament d'aplicacions multiplataforma. Creative Commons. Departament d'Ensenyament, Institut Obert de Catalunya. Dipòsit legal: B. 29430-2013. <https://ioc.xtec.cat/educacio/recursos>
- Alberto Oliva Molina. Acceso a datos. UD 1. Trabajo con ficheros XML. IES Tubalcaín. Tarazona (Zaragoza, España).

