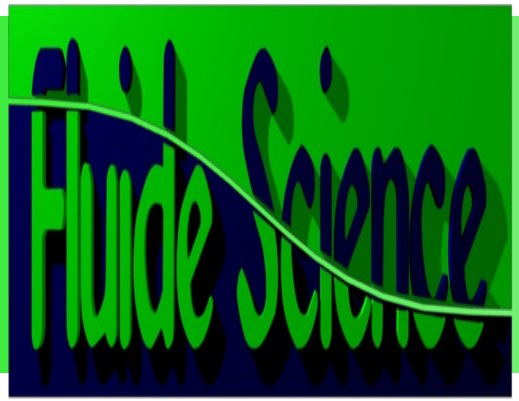
XML

Java

Course Goals

- Understand XML rules
- Use libraries
- How to create XML tree
- How to browse into XML tree

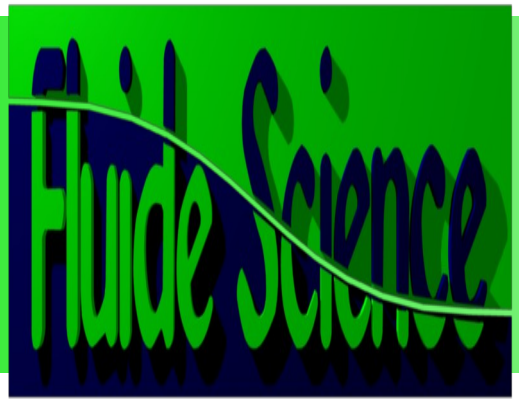


XML rules

XML rules

- Only one root element
- Everything opened must be closed (at correct place)
  - if a tag is self-sufficient, use <tag />
- Case sensitive !
- A tag can have :
  - some attributes (or not)
  - text

```
<bookstore>
        <owner name="ARNOLD" />
        <book category="COOKING">
                <title lang="en">Everyday Italian</title>
                <author>Giada De Laurentiis</author>
                <year>2005</year>
                <price>30.00</price>
        </book>
        <book category="CHILDREN">
                <title lang="en">Harry Potter</title>
                <author>J K. Rowling</author>
                <year>2005</year>
                <price>29.99</price>
        </book>
        <book category="WEB">
                <title lang="en">Learning XML</title>
                <author>Erik T. Ray</author>
                <year>2003</year>
                <price>39.95</price>
        </book>
</bookstore>
```



XML jars

XML jars

- You need to include in library path jdom-xxx.jar
- 2013 : last jar is jdom-2.0.5

XML creation

XML creation

- org.jdom2.Document class
  - constructor needs a “root” element
- org.jdom2.Element class
  - constructor needs a String (element's name)
  - addContent (Element) : to attach an element to another
  - setAttribute (Attribute) : to add an attribute to the element
  - setText (String) : to add some text
- org.jdom2.attribute
  - constructor needs a String name and a String value

XML creation

1) Create a root element “ing1promo2012”
2) attach it to the root element
3) Create document based on root element
4) Create an etudiant element “Etudiant” and attach it to root
5) Create an attribute [“code”, “SC01”] and attach it to the etudiant element
6) Create a nom element “nom”, set text “SCWARZENNEGGER” into it and attach nom element to etudiant.
7) Create a prenom element “prenom”, set text “Arnold” into it and attach nom element to etudiant.
8) At least create sexe and naissance children
9) goto 4 to a new student

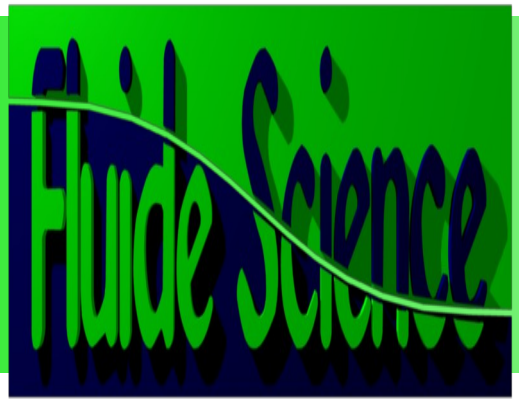
```
<?xml version="1.0" encoding="UTF-8"?>
<ing1promo2012>
    <etudiant code="SC01">
      <nom>SHWARZENNEGGER</nom>
      <prenom>Arnold</prenom>
      <sexe>masculin</sexe>
      <naissance>1947</naissance>
    </etudiant>
    <etudiant code="FO01">
      <nom>FOSTER</nom>
      <prenom>Jodie</prenom>
      <sexe>feminin</sexe>
      <naissance>1962</naissance>
    </etudiant>
</ing1promo2012>
```

XML output

XML output

```
public void affiche(OutputStream out) throws IOException
{
      XMLOutputter sortie=new XMLOutputter(Format.getPrettyFormat());
      sortie.output(document, out);
}
```

- we need document, so bring it to private attribute !
- to stdout
  - use affiche (System.out)
- to a file
  - use affiche (new FileOutputStream(String fichier))



your Job !

Job - 1

- Change tree in order to have next result :

```
<?xml version="1.0" encoding="UTF-8"?>
<ing1promo2012>
  <etudiants>
    <etudiant code="SC01">
      <nom>SHWARZENNEGGER</nom>
      <prenom>Arnold</prenom>
      <sexe>masculin</sexe>
      <naissance>1947</naissance>
    </etudiant>
    <etudiant code="FO01">
      <nom>FOSTER</nom>
      <prenom>Jodie</prenom>
      <sexe>feminin</sexe>
      <naissance>1962</naissance>
    </etudiant>
  </etudiants>
</ing1promo2012>
```

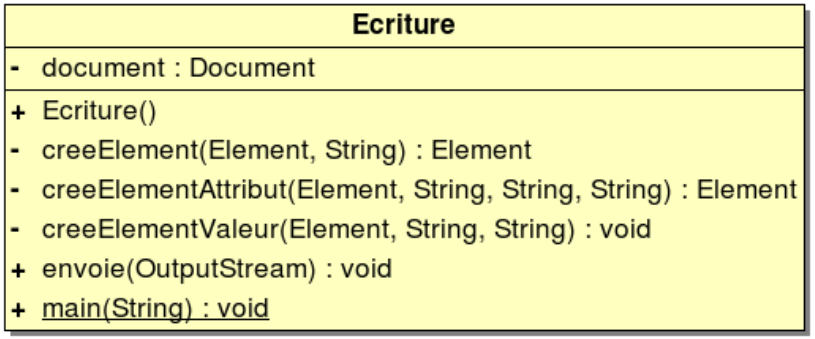
your second Job !

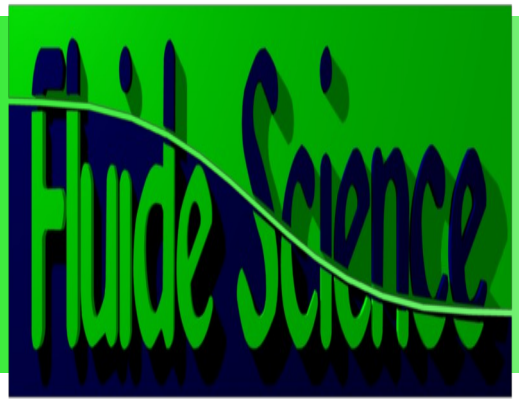
Job - 2

- Create a new class Ecriture simplifying xml writing

```
public Ecriture()
{
      Element root=new Element("ing1promo2012");
      document=new Document(root);
      Element etudiants,etudiant;

      etudiants=creeElement(root, "etudiants");

      etudiant=creeElementAttribut(etudiants, "etudiant", "code", "SC01");
      creeElementValeur(etudiant, "nom", "SHWARZENNEGGER");
      creeElementValeur(etudiant, "prenom", "Arnold");
      creeElementValeur(etudiant, "sexe", "masculin");
      creeElementValeur(etudiant, "naissance", "1947");

      etudiant=creeElementAttribut(etudiants, "etudiant", "code", "FO01");
      creeElementValeur(etudiant, "nom", "FOSTER");
      creeElementValeur(etudiant, "prenom", "Jodie");
      creeElementValeur(etudiant, "sexe", "feminin");
      creeElementValeur(etudiant, "naissance", "1962");
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ing1promo2012>
  <etudiants>
    <etudiant code="SC01">
      <nom>SHWARZENNEGGER</nom>
      <prenom>Arnold</prenom>
      <sexe>masculin</sexe>
      <naissance>1947</naissance>
    </etudiant>
    <etudiant code="FO01">
      <nom>FOSTER</nom>
      <prenom>Jodie</prenom>
      <sexe>feminin</sexe>
      <naissance>1962</naissance>
    </etudiant>
  </etudiants>
</ing1promo2012>
```

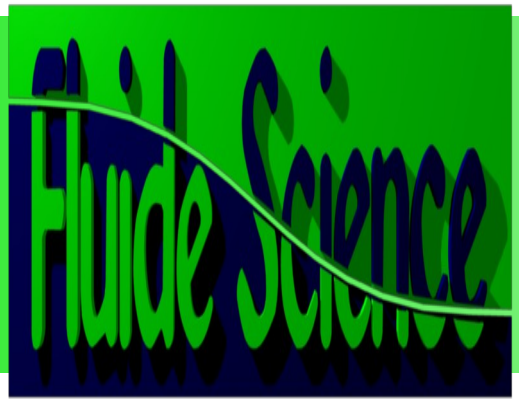




XML parsing

XML parsing

- org.jdom2.input.SaxBuilder class
  - constructor with no parameter
  - method build(new File(fichier)) : returns Document parsed
- org.jdom2.Document class
  - method getRootElement() : returns an Element (root !)
- org.jdom2.Element class
  - getChild(String elementName) : returns an Element
  - getChildren(String elementName) : returns a List of Elements
  - getAttributeValue(String attributeName) : returns a String containing its attribute value
  - getChildText(String elementName) : returns a String containing value of its child called “elementName”
- java.util.List
  - iterator() : returns an Iterator of this list
- java.util.Iterator
  - hasNext() : returns a boolean
  - next() : returns an Element of the list



your Job !

XML parsing

- Create a SaxBuilder instantiation.
- call SaxBuilder.build() method with xml file name and store result into a Document.
- retrieve root element.
- enter into root's child “etudiants”.
- List all etudiants' children and create an iterator.
- Loop
  - print attribute
  - print all gathered informations

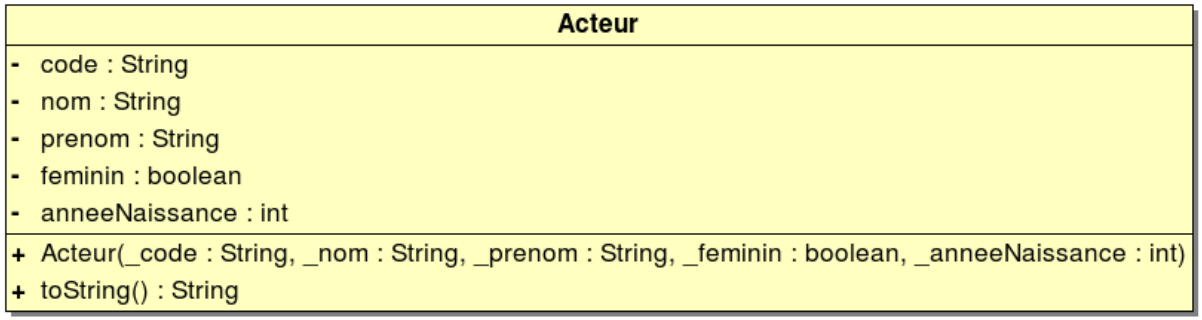
```xml
<?xml version="1.0" encoding="UTF-8"?>
<ing1promo2012>
  <etudiants>
    <etudiant code="SC01">
      <nom>SHWARZENNEGGER</nom>
      <prenom>Arnold</prenom>
      <sexe>masculin</sexe>
      <naissance>1947</naissance>
    </etudiant>
    <etudiant code="FO01">
      <nom>FOSTER</nom>
      <prenom>Jodie</prenom>
      <sexe>feminin</sexe>
      <naissance>1962</naissance>
    </etudiant>
  </etudiants>
</ing1promo2012>
```

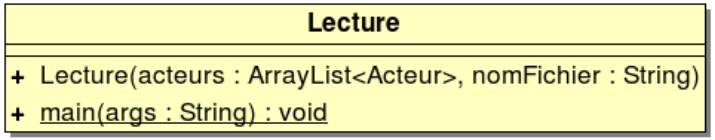
```
M. Arnold SHWARZENNEGGER né en 1947
Mme Jodie FOSTER née en 1962
```

your second Job !

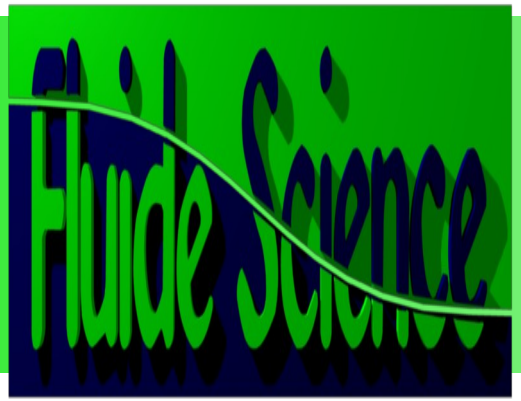
XML parsing & store into beans

- All informations will be stored into beans.
- Create a bean private fields, plain constructor, toString method
- In Lecture class :
  - Constructor will retrieve an arraylist (already instancied in main program), and an xml file name
  - During iterator's loop, it stores data in the bean collection.
  - main program :
    - declare an arraylist and instanciate it
    - call a new Lecture instanciation
    - print collection





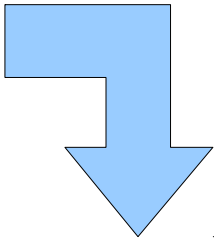

```
M. Arnold SHWARZENNEGGER né en 1947
Mme Jodie FOSTER née en 1962
```



pour les furieux fichiers DTD

DTD File

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT Fibonacci_Numbers (fibonacci*)>
<!ELEMENT fibonacci (#PCDATA)>
<!ATTLIST fibonacci index CDATA #IMPLIED>
```



Matches this xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Fibonacci_Numbers SYSTEM "fibonacci.dtd">
<Fibonacci_Numbers>
  <fibonacci index="1">1</fibonacci>
  <fibonacci index="2">2</fibonacci>
  <fibonacci index="3">3</fibonacci>
  <fibonacci index="4">5</fibonacci>
  <fibonacci index="5">8</fibonacci>
</Fibonacci_Numbers>
```

- Create a SaxBuilder instantiation.
- call SaxBuilder.build() method with xml file name and store result into a Document.
- retrieve root element.
- enter into root's child “etudiants”.
- List all etudiants' children and create an iterator.
- Loop
  - print attribute
  - print all gathered informations

XML Creation

```java
package com.ril14.xml;

import java.io.FileOutputStream;
import java.io.IOException;
import java.math.BigInteger;

import org.jdom2.*;
import org.jdom2.output.*;

public class EcrireXmlFibonacciDtd {

	public static void main(String[] args) {
		Element racine=new Element("Fibonacci_Numbers");
		DocType type=new DocType("Fibonacci_Numbers", "fibonacci.dtd");
		Document doc=new Document(racine, type);
		BigInteger low=BigInteger.ONE;
		BigInteger high=BigInteger.ONE.add(low);
		for(int i=1;i<=5;i++) {
			racine.addContent(new Element("fibonacci")
			.setAttribute(new Attribute("index", String.valueOf(i)))
			.setText(low.toString()));
			BigInteger temp=high;
			high=high.add(low);
			low=temp;
		}
		// la ligne suivante ferait planter la lecture
		// racine.addContent(new Element("toto"));
		try {
			XMLOutputter serial=new XMLOutputter(Format.getPrettyFormat());
			serial.output(doc, System.out);
			serial.output(doc, new FileOutputStream("fibonacci.xml"));
		} catch(IOException ioe) {
			System.err.println(ioe.getMessage());
		}
	}
}
```

XML Parsing

```
package com.ril14.xml;

import java.io.File;

import org.jdom2.Document;
import org.jdom2.input.SAXBuilder;
import org.jdom2.input.sax.XMLReaders;

public class LireXmlFibonacci {

	public static void main(String[] args) {
		SAXBuilder sxb=new SAXBuilder(XMLReaders.DTDVALIDATING);
		try {
			Document doc=sxb.build(new File("fibonacci.xml"));
		} catch (Exception e) {
			System.err.println(e.getMessage());
		}
	}
}
```


Fluid Science