# TD3 – NF28 : parsing SAX

## *Présentation*

### Objectif

L'objectif de ce TD est de réaliser le parsing SAX d'un fichier XML dont la structure est donnée. Deux principaux parsing existent : le parsing DOM (Document Object Model) et le parsing SAX. Le parsing DOM réalise un arbre DOM en mémoire représentant le fichier XML. Le parsing SAX laisse toute liberté quant au résultat du parsing. Il permet de construire ici une structure qui ne nécessite qu'un unique passage sur le fichier. La structure arborescente créée ici servira de modèle à un objet graphique de type JTree.

### Exemple de fichier XML à parser

```
<?xml version="1.0" encoding="UTF-8"?>
<contacts>
  <amis/>
  <collegues>
    <contact>
     <nom>Claude Moulin</nom>
     <mail>claude.moulin@utc.fr</mail>
     <icone>C:/data/nf28_06/gui/images/image3.jpg</icone>
    </contact>
    <contact>
     <nom>Dominique Lenne</nom>
     <mail>dominique.lenne@utc.fr</mail>
     <icone>C:/data/nf28_06/gui/images/image6.jpg</icone>
    </contact>
  </collegues>
  <famille></famille>
</contacts>
```

Ce fichier représente un ensemble de contacts. Ils sont classés par catégories. Une catégorie peut être vide (<amis/> ou <famille></famille>). Un contact peut ne pas être encore classé ; il apparaît alors en dehors des catégories avant la première ou après la dernière. Seule la structure XML d'un contact est figée. Elle est contenue entre les balises <contact> et </contact> et contient les éléments nom, mail et icône. Son parsing engendre les nœuds de plus bas niveaux de l'arbre.


**Figure 1 : Vue des contacts dans un JTree obtenue après le parsing SAX**

## *Proposition de phases de réalisation*

### Phase 1 : Réalisation du parsing

L'objectif du TD étant également de préparer un modèle pour un JTree, le parsing SAX doit permettre de créer un arbre faisant directement appel à des objets de type Contact.
Java fournit une classe nœud, DefaultMutableTreeNode implémentant l'interface TreeNode à travers l'interface MutableTreeNode, et possédant toutes les caractéristiques nécessaires à la création d'un arbre. On peut soit spécialiser cette classe pour créer des nœuds adaptés au problème traité, soit utiliser directement cette classe et associer à chaque nœud de l'arbre un objet utilisateur particulier. On choisira cette dernière solution ici.

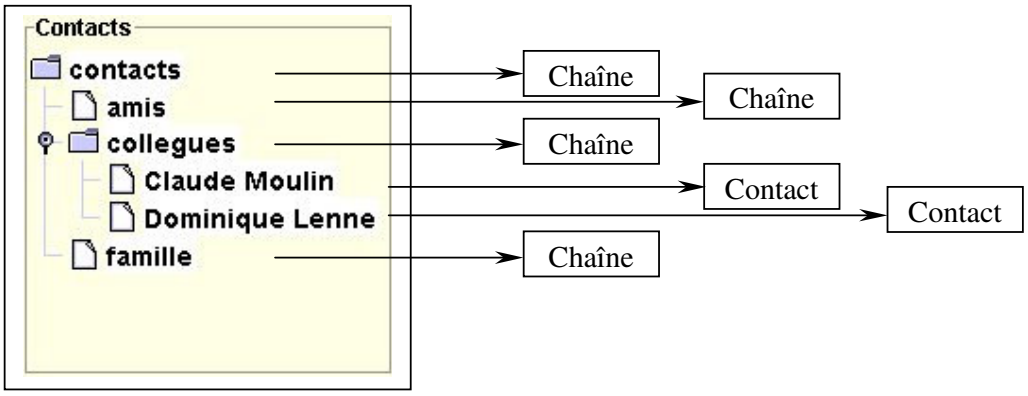


**Figure 2 : Objets associés aux nœuds**

La figure 2 montre les objets associés aux nœuds catégorie et aux nœuds contact. Il suffit de créer une méthode toString dans les classes concernées renvoyant la chaîne qui doit être affichée dans la vue.
Créer une classe Contact possédant les trois propriétés pour le nom, le mail et l'icône.
Créer une classe ContactHandler héritant de DefaultHandler et compléter les méthodes de réponses aux événements SAX. Elles ont pour but de créer un arbre composé de nœuds de type DefaultMutableTreeNode.
Mettre en place le parsing à partir du nom d'un fichier. On pourra créer à ce propos une classe ContactFacility et une méthode parse.

```
Public static void parse(java.lang.String filename)
   throws SAXException, FileNotFoundException, IOException {
   //String xmlReaderClassName = "org.apache.crimson.parser.XMLReaderImpl";
   String xmlReaderClassName = "org.apache.xerces.parsers.SAXParser";
   XMLReader xr = XMLReaderFactory.createXMLReader(xmlReaderClassName);
   ContactHandler handler = new ContactHandler();
   xr.setContentHandler(handler);
   xr.setErrorHandler(handler);
   FileReader r = new FileReader(filename);
   xr.parse(new InputSource(r));
}
```

Les librairies java 5.0 et 6, n'intègrent plus de parsers XML comme les versions précédentes. Il faut donc télécharger et installer dans Eclipse un fichier .jar contenant les classes correspondantes. La première ligne de la méthode *parse* est à adapter en conséquence. On pourra choisir par exemple le parser Xerces, que l'on peut trouver sur le site d'Apache[1] et sur le site NF28. Le fichier archive a pour nom xercesImp.jar

[1] http://xerces.apache.org/xerces-j/

## Phase 2 : Vérification de la structure d'arbre créée

Créer une classe ContactTreeModel héritant de DefaultTreeModel pour servir de modèle au JTree chargé d'afficher les contacts. Elle possède un constructeur à partir d'un TreeNode. Ajouter une méthode getContactTreeModel dans la classe ContactHandler renvoyant un ContactTreeModel créé à partir du nœud racine de l'arbre issu du parsing d'un fichier XML. Modifier la méthode parse de la classe ContactFacility pour qu'elle retourne un ContactTreeModel.

```
public static ContactTreeModel parse(java.lang.String filename) throws
                    SAXException, FileNotFoundException, IOException {
  …
  return handler.getContactTreeModel();
}
```

Ajouter une méthode toXML dans la classe ContactTreeModel renvoyant une chaîne XML correspondant à la représentation XML de l'arbre présent dans cette structure.
Tester le parsing Sax.

## Phase 3 : Interface Graphique

Réaliser une vue faisant apparaître le JTree auquel est associé le modèle issu du parsing d'un fichier XML d'un ensemble de contacts, et une zone de texte montrant la génération de la chaîne XML à partir de l'arbre construit lors du parsing. La Figure 3 montre la visualisation de l'arbre des contacts et la visualisation XML de l'arbre affiché. Le panneau de gauche est créé indépendamment (ContactTreePanel) et inséré dans la fenêtre de l'application.

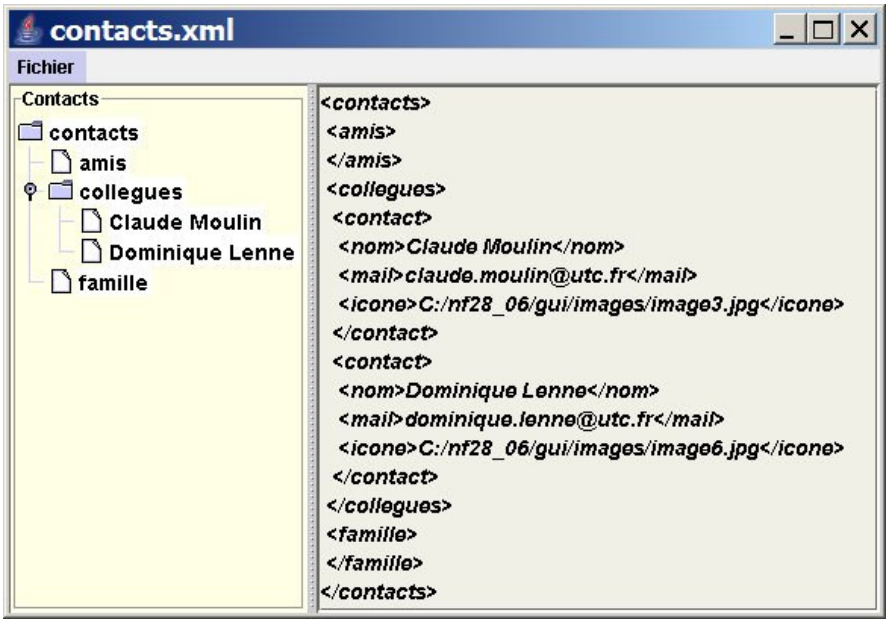


**Figure 3 : Interface de l'application**