On considère le fichier XML suivant :

```
<etudiants>
      <etudiant nom="Durand" prenom="Emile" />
      <etudiant nom="Dupont" prenom="Micheline" />
</etudiants>
```

1. Ecrire une application GestionEtudiant en utilisant le package org.w3x.dom du jdk de sun qui a les fonctionnalités suivantes :
   a. `Document chargerXML(String fileName)`
   On utilisera la classe DocumentBuilderFactory pour fabriquer un parser DocumentBuilder qui vous permettra de charger le document XML. On peut optimiser le parser en réglant les propriétés de la fabrique.

   Corrigé (le + simple) :

   ```
   DocumentBuilderFactory fabrique = DocumentBuilderFactory.newInstance();
   DocumentBuilder parser = fabrique.newDocumentBuilder();
   return parser.parse(new File(fileName));
         + exceptions +
   ```

   b. `void afficherEtudiants(Document listeEtudiants)`
   Parcours la liste des étudiants et affiche le nom et prénom de chacun.

   Corrigé : 2 méthodes
   1- parcourir l'arbre avec getDocumentElement puis getChildNodes jusqu'à tomber sur les étudiants (attention aux fils de type texte qui viennent se glisser avec l'indentation)
   2- plus simple : getElementsByTagname sur le nom de balise "etudiant"

   c. `void modifierEtudiant(Element etudiant, String newNom, String newPrenom)`
   Modifie un étudiant dont on connait la reference objet dans l'arbre XML avec un nouveau nom et un nouveau prénom.
   Corrigé : utiliser directement la méthode setAttribute de l'élément

   d. `void ajouterEtudiant(Document listeEtudiants, String newNom, String newPrenom)`
   Ajoute un nouvel étudiant en créant un nouveau noeud dans l'arbre XML.
   Corrigé : créer le nouveau nœud à partir du document puis l'attacher à l'arbre comme fils de l'élément "etudiants"

   e. `void sauverXML(Document listeEtudiants, String fileName)`
   Sauve l'arbre XML en mémoire vive dans un fichier XML. De la même façon que pour le chargement, on utilisera un TransformerFactory pour fabriquer un Transformer capable de réaliser la sauvegarde. On fixera les propriétés suivantes - définies dans la classe OutputKeys - sur le Transformer :
   - V l'encodage
   - V l'indentation
   - V la méthode (xml)

- V l'ajout de déclaration d'entête XML

Corrigé :

```
TransformerFactory tf = TransformerFactory.newInstance();
Transformer transformer = tf.newTransformer();
transformer.setOutputProperty(OutputKeys.METHOD, "xml");
transformer.setOutputProperty(OutputKeys.INDENT, "yes");
transformer.setOutputProperty(OutputKeys.OMIT_XML_DECLARATION, "no");
File f = new File(filename);
if (f.exists()) {
     f.delete();
}
StreamResult nvFichier = new StreamResult(f);
transformer.transform(new DOMSource(arbreDOM), nvFichier);
        + exceptions +
```

f. un `main` qui charge un fichier donné en 1er argument de l'appli, affiche la liste des étudiants, modifie le 1er étudiant en Dupont Maurice, ajoute l'étudiant Dumoulin Valentine et sauve l'arbre résultat dans le fichier de nom passé en 2ème argument de l'appli.

2. Refaire l'application précédente avec JDom

   Corrigé : à suivre

3. Rajouter la validation par rapport à un schéma XSD au chargement du fichier XML :

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="etudiants ">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="etudiant" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:complexContent>
              <xs:restriction base="xs:anyType">
                <xs:attribute name="nom" type="xs:string"/>
                <xs:attribute name="prenom" type="xs:string"/>
              </xs:restriction>
            </xs:complexContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Il faut faire les réglages suivants sur la fabrique de parser :

- V prise en compte des espaces de nom : non
- V validation : oui
- V ajouter l'attribut `JAXP_SCHEMA_LANGUAGE` avec la valeur `W3C_XML_SCHEMA`
- V ajouter l'attribut `JAXP_SCHEMA_SOURCE` avec comme valeur le fichier XSD (de type `File`)

avec les constantes :

- V `JAXP_SCHEMA_LANGUAGE` : `"http://java.sun.com/xml/jaxp/properties/schemaLanguage"`
- V `W3C_XML_SCHEMA` : `"http://www.w3.org/2001/XMLSchema"`
- V `JAXP_SCHEMA_SOURCE` : `"http://java.sun.com/xml/jaxp/properties/schemaSource"`

Pour pouvoir suivre le déroulement de la validation on associe un error handler ; on pourra utiliser la classe `DefaultHandler` ou définir une spécialisation de celle-ci pour s'approprier le traitement d'erreurs.

Corrigé : voir code en annexe (définir son propre handler n'est pas obligatoire, cela permet de récupérer l'état de la validation et de gérer son système de message (affichage ou log)).

4. Refaire les exercices 1 à 3 en gérant les espaces de noms :

```
<exo:etudiants xmlns:exo="http://ing2.eisti.fr/technoXML">
      <exo:etudiant nom="Durand" prenom="Emile" />
      <exo:etudiant nom="Dupont" prenom="Micheline" />
</exo:etudiants>
```

Corrigé :

Utiliser les fonctions avec suffixes NS, par exemple getElementsByTagnameNS. Attention le 1er paramètre n'est pas le prefixe ("exo") mais l'espace de nom ("http://ing2.eisti.fr/technoXML").

Annexe : validation d'un document XML avec un schéma avec DOM

```
import javax.xml.parsers.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import org.w3c.dom.*;
import java.io.*;

class XmlValidHandlerManager extends DefaultHandler {

      private boolean bool_isValid = true;

      public void reportState() {
            if (isValid()) {
                  System.out.println("Document valide");
            } else {
                  System.out.println("Document non valide");
            }
      }

      public boolean isValid() {
            return bool_isValid;
      }

      private void reportProblem(String generalMessage, SAXParseException e) {
            System.err.println(generalMessage);
            System.err.println("   Ligne   : "+e.getLineNumber());
            System.err.println("   Colonne : "+e.getColumnNumber());
            System.err.println("   Message : "+e.getMessage());
      }

      public void fatalError(SAXParseException e) {
            reportProblem("Erreur de validation - Erreur fatal",e);
            bool_isValid = false;
      }

      public void error(SAXParseException e) {
            reportProblem("Erreur de validation - Erreur",e);
            bool_isValid = false;
      }

      public void warning(SAXParseException e) {
            reportProblem("Erreur de validation - Warning",e);
```

```
            bool_isValid = false;
      }
}

public class Validator {

      static final String JAXP_SCHEMA_LANGUAGE =
            "http://java.sun.com/xml/jaxp/properties/schemaLanguage";

      static final String W3C_XML_SCHEMA = "http://www.w3.org/2001/XMLSchema";

      static final String JAXP_SCHEMA_SOURCE =
            "http://java.sun.com/xml/jaxp/properties/schemaSource";

      // valider un document xml avec un schema sans prise en compte des espaces
      // de matière
      public static Document domParseWithXsd(String xmlFileName, String xsdFileName) {
            return domParseWithXsd(xmlFileName, xsdFileName, false);
      }

      public static Document domParseWithXsd(String xmlFileName, String xsdFileName,
                  boolean isNameSpaceAware) {
            try {
            DocumentBuilderFactory dbFactory = DocumentBuilderFactory.newInstance();
            DocumentBuilder db;
            Document dom;
            XmlValidHandlerManager errorHandler = new XmlValidHandlerManager();
            // Reglages de la fabrique
            dbFactory.setIgnoringComments(true);
            dbFactory.setNamespaceAware(isNameSpaceAware);
            dbFactory.setValidating(true);
            dbFactory.setAttribute(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
            dbFactory.setAttribute(JAXP_SCHEMA_SOURCE, new File(xsdFileName));
            db = dbFactory.newDocumentBuilder();
            db.setErrorHandler(errorHandler);
            dom = db.parse(new File(xmlFileName));
            errorHandler.reportState();
            return dom;
            } catch (ParserConfigurationException e) {
                  System.err.println(e.getMessage());
                  return null;
            } catch (SAXException e) {
                  System.err.println(e.getMessage());
                  return null;
            } catch (java.io.IOException e) {
                  System.err.println(e.getMessage());
                  return null;
            }
      }


      public static void main(String[] args) {
            if (args.length > 1) {
                  // args[0] : XML
                  // args[1] : XSD
                  domParseWithXsd(args[0], args[1]);
            }
      }
}
```