

Document de Conception

**Membres du groupe**

**NOUPELAH VLADIMIR**

**NYASSA NOAH**

**TANWI NKIAMBOH**

**KOUAYEP PATERNE**

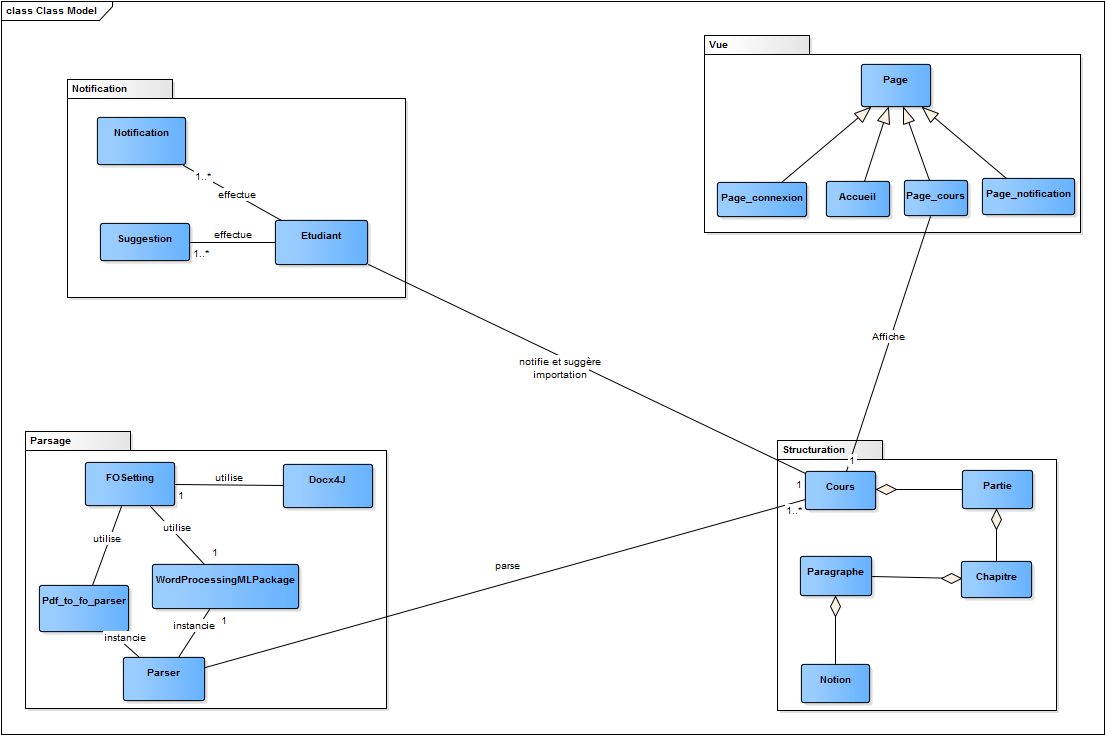
**TCHUITSE DJOMO**

# DIAGRAMME DE CLASSES

Les différentes classes sont groupées en fonction des modules :

* **La vue** : l’accueil, la page de cours (le cours actuellement utilisé), la page qui liste tous les cours disponibles, la page de notifications
* **Le parseur**: parser (classe principale), Docx4J et WordProcessingMLPackage qui sont des bibliothèques de traitement de documents Word, FO Setting, PDF\_to\_fo\_Parser qui est la bibliothèque de traitement de documents PDF.
* **La structuration** : les cours, les parties, les chapitres, les paragraphes, les notions
* **La notification** : les étudiants, les notifications, les suggestions.

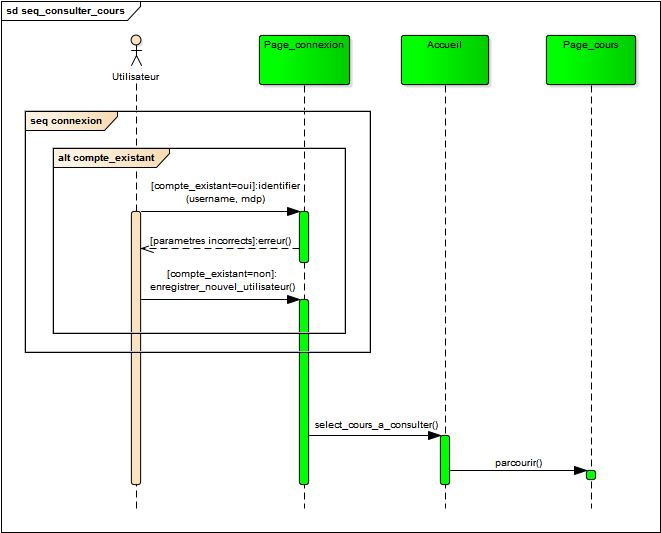
Le diagramme de classe est donc le suivant :



# DIAGRAMME DE SEQUENCE

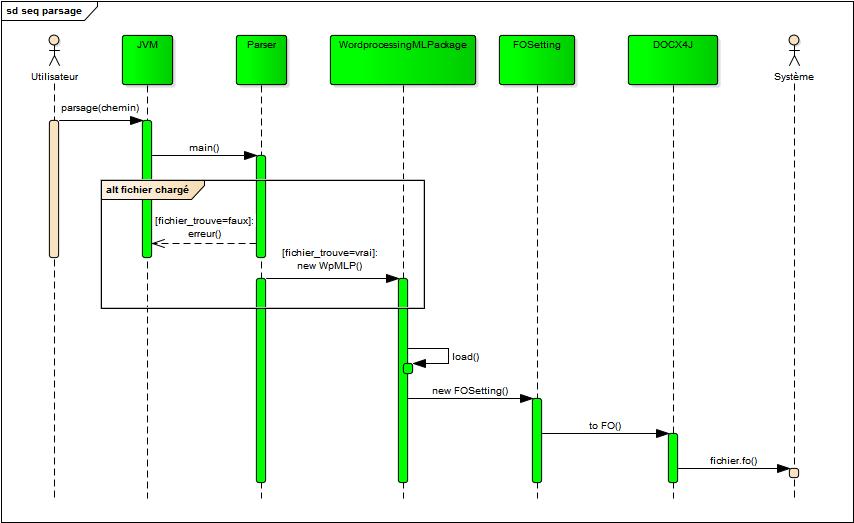
Les principales séquences dans l’utilisation du module sont :

## La consultation d’un cours déjà importé et préalablement structuré



L’importation d’un nouveau d’un nouveau cours, ce qui passe par les processus de parsage et de structuration.

## Le parsage d’un cours Word



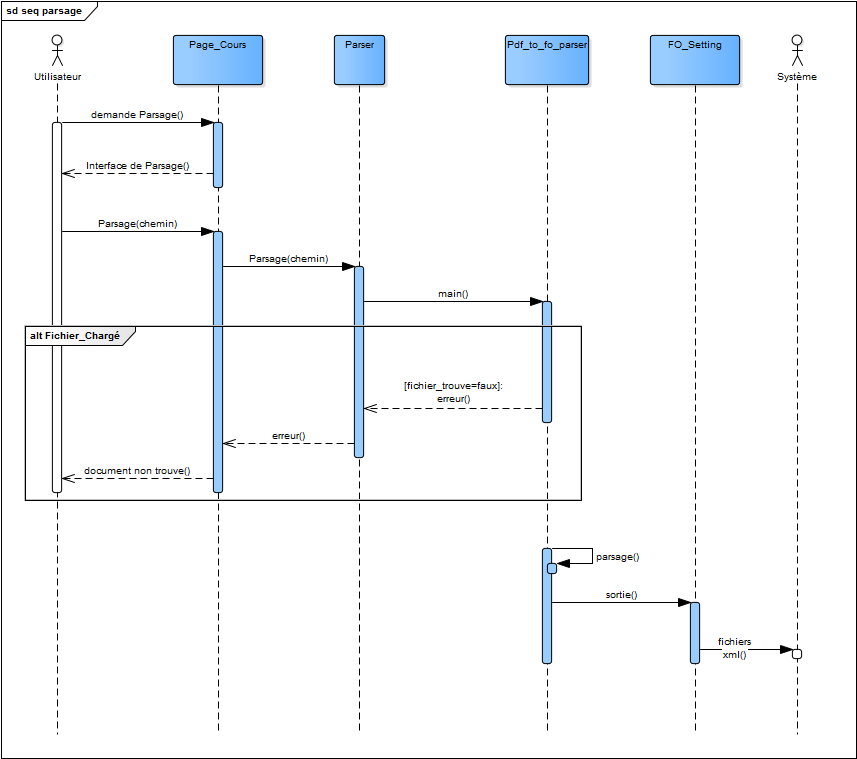
Le processus de conversion d’un fichier du format docx vers le format xslfo se fait suivant le processus suivant :

* L’utilisateur appelle le programme de parsage (fichier jar) en lui passant en paramètres le chemin vers le fichier source à convertir (fichier au format docx) :

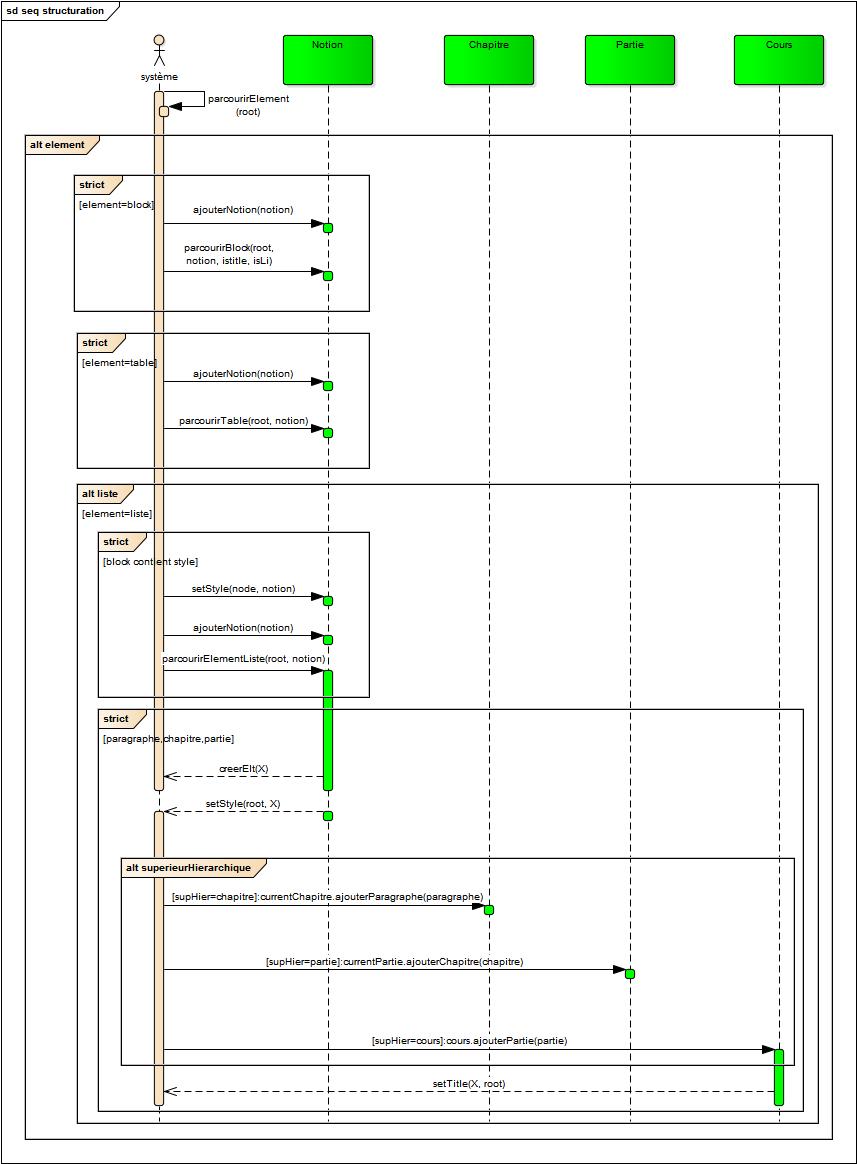
**Java –jar Docx\_To\_Fo\_Parser.jar <fichier source au format docx>**

* La machine virtuelle Java appelle la méthode main de la classe **Parser**
* La méthode main appelée lit et vérifie le paramètre de la méthode et renvoie un message d’erreur si elle n’arrive pas à charger le fichier.
* Si le fichier est bien présent et accessible en lecture, la méthode main crée et instancie un objet de la classe **WordprocessingMLPackage**. Le fichier docx (son flux qui a été lu) est chargé dans l’objet de la classe **WordprocessingMLPackage** grâce à la méthode **load ()** de cette classe.
* Le main crée ensuite un objet de la classe FOSettings et l’instancie avec des valeurs par défaut accessibles par une méthode statique de la classe **DOCX4J** (**createFOSettings ()**) ;
* D’autres paramètres sont ajoutés à l’objet de la classe FOSettings créé tels que le nom du fichier de sortie, son extension et bien d’autres.
* La méthode main crée ensuite un flux de sortie correspondant au fichier au format xslfo qui sera produit en sortie par le parseur.
* Le main appelle ensuite la méthode statique **toFO ()** de la classe **DOCX4J** en lui passant en paramètres le flux de sortie et l’objet de la classe FOSettings contenant les paramètres de conversion.
* Cette méthode réalise la conversion en affichant les différentes étapes et les éventuelles erreurs en console. A la fin du processus un fichier contenant le même nom que le fichier source augmentée d’une « .**fo** » à la fin est produit dans le même répertoire que le fichier source.
* Ce fichier est formaté en xslfo

## Le parsage d’un cours PDF



## La structuration



* On parcours tous les nœuds qui ne sont pas des métadonnées: parcoursElement(Node root)
* On écrit ensuite les documents de descriptions

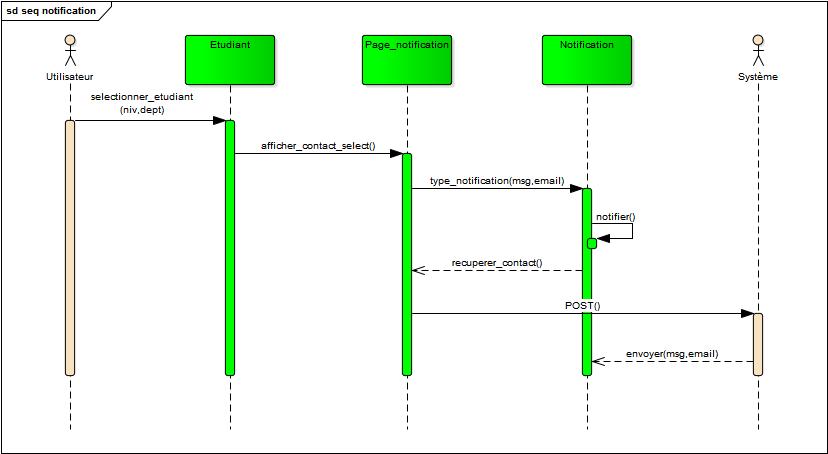
Description de la fonction parcoursElement(Node root)

* S'il s'agit d'un fo:block je creer une Notion
* On ajoute la notion: ajouteNotion(Notion notion)
* On parcourt l'élément: parcoursBlock(Node root, Notion notion, boolean isTitle, boolean isLi)
* S'il s'agit d'un fo:table je creer une Notion
  + On ajoute la notion: ajouteNotion(Notion notion)
  + On parcourt la table: parcoursTable(Node root, Notion notion)
* S'il s'agit d'un élement de liste (un fo:list-block)
* Si son fo:list-item-label a un fo:block de font-family symbol, je creer une notion
  + - On copie le style: setStyle(Node node, Element notion)
    - On ajoute la notion: ajouterNotion(Notion notion)
    - On récupère le contenu de l'élément de liste: pacoursElementListe(Node root, Notion notion)
* Si non, il s'agit soit d'un paragraphe, un chapitre ou une partie

*notons "X" qui sera un paragraphe, un chapitre ou une partie*

* + - On cree un élément X,
    - on met son style : setStyle(Node root, X x)
    - On l'ajoute à son supérieur hiérarchique (chapitre, partie ou cours): cours.ajouterPartie(Partie partie), currentChapitre.ajouterParagraphe(Paragraphe paragraphe) ou currentPartie.ajouterChapitre(Chapitre chapitre).
    - On met sont titre: setTitle(X x, Node root)

## La notification



# SOURCES

Nous nous sommes inspirés de la version précédente du document de conception du module XCSM pour réaliser les améliorations.