SER – Laboratoire 2 – Parsing XML avec JDOM 2

Objectifs

- Comprendre la structure d'un document XML
- Lecture d'un fichier XML
- Écriture dans un autre format de sortie

But

Ce laboratoire fait suite au laboratoire 1 sur les parties d'échecs.

Un document XML (**tournois_fse.xml**) est fourni contenant plusieurs parties d'échecs avec un seul tournoi (mais il pourrait y en avoir plusieurs…). On vous fournit également la DTD correspondante ce qui vous permettra de mieux comprendre sa structure.

On vous demande de faire un programme java qui va parser ce fichier et produire plusieurs fichiers de sortie :

- Chacun de ces fichiers de sortie sera au format PGN (qui est un format standard de codage de parties d'échecs) et correspondra à une des parties présentes dans le fichier XML

- Les informations que vous devez stocker dans chacun de ces fichiers de sorties (1 fichier par partie) sont **uniquement** les coups joués dans la partie correspondante.

Chaque fichier PGN qu'on vous demande de produire va rester très simple et doit respecter la structure suivante :

1 cb1 cn1
2 cb1 cn2
…
N cb1 cnN

- La valeur indiquée en **vert** est le numéro **n** du « tour » joué (on change de numéro à chaque fois que les blancs **et** les noirs ont joué un coup).

- La valeur indiquée en **bleu** est la notation du coup des blancs dans le tour **n**

- La valeur indiquée en **violet** est la notation du coup des noirs dans le tour **n**

Notation à utiliser pour les coups

Tout d'abord, chaque pièce du jeu d'échecs est identifiée par une lettre (nous allons utiliser ici le format anglais) :

- Q = Dame
- K = Roi
- R = Tour
- N = Cavalier
- B = Fou
- *A noter qu'il n'y a pas de lettre pour le pion*

Quelques exemples pour bien comprendre :

- Le cavalier s'est déplacé en g8 → **N**g8
- Le fou qui se trouvait sur la case a3 s'est déplacé en c5 → B**a3c5**
- La dame s'est déplacée en d6 et a éliminé une pièce adverse → Q**x**d6
- La tour qui se trouvait sur la case b8 s'est déplacée en g8 et a éliminé une pièce adverse → Rb8xg8
- Le pion se déplace sur la case a4 → a4
- Le pion qui se trouvait sur la case a4 s'est déplacé en b5 et a éliminé une pièce adverse → **a**xb5
    - A noter que si c'est un pion qui élimine une pièce, on indiquera **toujours** la colonne de départ *(le numéro de ligne étant inutile dans ce cas-là dans le cas où les règles normales du jeu d'échecs sont appliquées…)*

On peut donc en déduire la structure suivante : Piece_CaseDepart_**Elimination**_**CaseArrivee**

Cas particuliers :

- Lorsque c'est un **pion** qui **élimine** une pièce, alors la case de départ doit toujours être fournie (pour qu'on puisse indiquer la **colonne de départ**) (voir le dernier coup listé ci-dessus)
- Le petit roque se note de la manière suivante : O-O
- Le grand roque se note de la manière suivante : O-O-O

- La promotion (qui concerne uniquement le déplacement d'un pion) se note de la manière suivante :
    - Le pion est arrivé sur la case f8 et est promu en Dame → f8=**Q**
        - *La lettre après le = indique la pièce qui a été choisie pour la promotion (Q pour Dame, B pour Fou, R pour Tour, N pour Cavalier)*

- *Si le coup donne lieu à un échec, alors on rajoute un + par exemple :*
    - *Le cavalier se déplace en c7 et fait échec au roi : Nc7+*

- *Si le coup donne lieu à un échec et mat, alors on rajoute un # par exemple :*
    - *Le cavalier se déplace en c7 et fait échec et mat : Nc7#*

Voilà comment vous devrez « traduire » chaque coup dans une partie dans le fichier final concerné (tous cas possibles qu'on vous demande de traduire ont été présentés ci-dessus).

Travail à faire

Partie 1 – Transformation d'un coup dans la notation PGN

On vous fournit des classes (paquetage ch.heigvd.ser.labo2.**coups**) que vous devrez compléter concernant la conversion des coups dans le format PGN (vous devez modifier **uniquement** les parties où il est indiqué) :

- **// TODO : A implémenter**
  Et rien d'autre !

On vous a fourni des tests automatisés que vous devrez lancer pour vérifier votre implémentation concernant cette première partie. La procédure vous est expliquée pendant la présentation du labo.

Partie 2 – Parsing avec JDOM 2

Vous devez dans le fichier Main.java parser le document **tournois_fse.xml** fourni et produire les fichiers de sorties demandé.

Très important : On vous impose d'utiliser JDOM 2 et de ne pas lire le fichier XML à la main !

Pour l'écriture dans les fichiers de sortie, on vous conseille d'utiliser la class **PrintWriter** qui s'utilise comme ceci :

- PrintWriter pw = new PrintWriter(new FileWriter(fileName));

Et vous pouvez utiliser les méthodes print() / println() qui vous seront très utiles.

Partie 3 – Utilisation d'un simulateur de partie d'échecs

Ce n'est pas un travail à faire, mais simplement pour tester le résultat de vos fichiers de sortie :

- Allez sur le site : https://www.chess.com/fr/analysis
- Cliquez ensuite sur Commencer
- Ensuite à droite vous avez un menu « Charger le PGN », copier-collez le résultat d'un des fichiers de sortie à l'intérieur
- Après cela, vous pouvez « jouer » la partie correspondante avec les boutons de navigation qui se trouvent en bas…

Exemple :

Imaginons que le fichier XML contienne la partie suivante :

```
1 e4 e5
2 Bc4 Bc5
3 Qh5 Nc6
4 Qxf7#
```

Après avoir inséré la partie dans le simulateur et **après avoir navigué jusqu'au dernier coup** :

Contraintes

Voici différentes contraintes à respecter :

- Ne faites pas d'avantage de travail que ce qui est demandé pour la notation PGN (car il y a bien d'autres choses qu'on peut indiquer dans ces fichiers).
    - *Implémenter des choses qu'on ne vous demande pas ne sera pas payé…*

- Il est **interdit** de modifier le fichier XML fourni, ni d'utiliser votre propre fichier du labo 1

- Il est **interdit** de modifier le code fourni, il faut uniquement toucher les parties où il est indiqué : A implémenter

- Vous **devez** utiliser la librairie Jdom2 version 2.0.6

- Les classes fournies se trouvent dans le paquetage ch.heigvd.ser.labo2. Il est **interdit** de modifier le paquetage de ces classes.

On vous rend attentif que vous serez fortement pénalisé si l'une de ces contraintes n'est pas respectée.

Code fourni (projet Maven)

On vous fournit un projet Maven sur le repos Cyberlearn qui contient déjà les classes à compléter, ainsi que les dépendances nécessaires pour la librairie JDOM2 (indiquée dans le fichier pom.xml). Maven pour information est un outil de déploiement d'applications Java assez poussé.

A noter que dans le cadre de ce labo, on l'utilise uniquement pour les tests automatisés et la dépendance avec JDOM2 version 2.0.6. Ce qui nous simplifiera la tâche.

Vous pouvez importer ce qu'on vous a fourni dans un projet IntelliJ, ou tout IDE supportant Maven. Faites attention dans l'importation du projet d'indiquer qu'il s'agit d'un projet Maven.

A rendre

Cette fois-ci il n'y a **pas de rapport à rendre**.

Vous devez cependant fournir les éléments suivants :

- Le code java complet
- Les 4 fichiers de sorties correspondant aux 4 parties qui se trouvent dans le document XML fourni
- 4 captures d'écran représentant les états finaux de chacune de ces parties de l'échiquier graphique (que vous aurez obtenu avec le simulateur indiqué)

Rendu

Vous devez réaliser ce laboratoire par groupe de 3 personnes.

Vous avez 4 périodes à disposition pour la réalisation de ce laboratoire.

Votre travail doit être rendu dans une archive ZIP (format : Nom1_Nom2_Nom_3_SER_Labo_2.zip) sur Cyberlearn avant le **dimanche 14 avril 2019 à 23h55**.

Bon travail !