Université Paris Diderot **Programmation Orientée Objet**
L2 **Année 2016-2017**

TP n°8

Exceptions et classes internes

Arborescence de fichiers

Le but de cette partie est de réaliser un programme capable d'afficher sous forme d'arborescence le contenu d'un répertoire de fichiers. Pour ce faire, nous aurons besoin des classes `File` et `FileNotFoundException`, qui se trouvent dans le package `java.io`.

Nous allons commencer par créer une classe `Arbre` qui représente le contenu d'un répertoire.

Exercice 1 Le modèle
Écrire la classe `Arbre`, avec les attributs suivants :

- Une classe interne `Noeud`, qui représente un noeud de l'arborescence (un fichier ou un répertoire), et contiendra un champ `boolean repertoire` qui indique si c'est un répertoire ou non, un champ `nom` et un champ `taille`. On ajoutera également un champ `fils` de type `ArrayList<Noeud>` qui représente les fichiers contenus dans ce noeud, si ce noeud est un répertoire (il sera à `null` dans le cas contraire).
- Un attribut `Noeud racine`, le répertoire représenté par cet arbre

Exercice 2 Création de l'arbre
Ajouter à la classe `Noeud` un constructeur qui prend un paramètre un objet `File` et qui effectue les opérations suivantes :

- Si le fichier correspondant n'existe pas, lever une `FileNotFoundException`
- Sinon, initialiser les champs `nom`, `taille` et `repertoire`. Si le fichier concerné est un répertoire, alors on initialise le membre `fils` et on le remplit récursivement. Sinon, `fils` est laissé à `null`.

On ajoutera enfin un constructeur à la classe `Arbre` qui prend en paramètre une chaîne de caractères représentant le chemin de la racine de l'arbre. Si le chemin n'existe pas, on lèvera une `FileNotFoundException`.

Aide : Consulter la javadoc de la classe `File` pour plus d'informations. On se servira notamment des méthodes `exists()`, `getName()`, `length()` et `listFiles()`.

Exercice 3 Affichage
Écrire une méthode `void afficher()` dans la classe `Arbre`, qui affiche l'arbre de la manière suivante :

```
racine (15)
  fichier1 (100)
  fichier2 (200)
  rep1 (200)
    fichier3 (0)
    fichier4 (5)
  rep2 (100)
    rep3 (100)
     fichier5 (100)
```

Où chaque noeud est affiché à sa profondeur, avec son nom suivi de sa taille entre parenthèses

Classes internes dynamiques et statiques : un petit jeu

Règles du jeu Le jeu des serpents et des échelles est un jeu de plateau avec des cases numérotées de 1 à n. Sur ce plateau sont aussi disposées des échelles et des serpents. Tous deux vont d'une case à une autre. Les échelles permettent d'aller de la case du numéro le plus petit à celui de numéro le plus grand (on "monte"), et les serpents obligent à "redescendre" de la case de numéro le plus grand à celui de numéro le plus petit.

Pour une image voir `http://nl.wikipedia.org/wiki/Slangen_en_ladders`

Le jeu se joue à plusieurs ; chaque joueur a un pion qui le représente. Au début, tous les pions sont sur la case 1 ; le but étant d'arriver le premier à la case n. À chaque tour, chaque joueur lance un dé et avance du nombre de cases indiqué par le dé. Lorsque, après avoir avancé, un joueur se trouve au bas d'une échelle ou en haut d'un serpent, il est obligé de l'emprunter.

Si un joueur arrive à la case finale avant d'avoir épuisé tous les points de son dé, il doit reculer.

Exercice 4 Programmation

On définira une classe `Jeu` qui contiendra une classe interne statique `Case` et une classe interne dynamique (c'est-à-dire non statique) `Joueur`. Par ailleurs, la classe `Jeu` contiendra un tableau de `Case` (le plateau) et un tableau de `Joueur`.

Une case aura comme attribut le numéro de case vers où l'on va si on s'y arrête, éventuellement elle-même. Si par exemple, il y a une échelle entre les cases 3 et 15 et rien d'autre à la case 15, la case 3 contiendra l'entier 15, et la case 15 contiendra 15. La classe `Case`, contiendra aussi une méthode `int destination()`.

Un joueur lui saura son numéro (pour simplifier l'affichage) ainsi que le numéro de la case où il est, et contiendra entre autres une méthode `boolean joueUnTour()` qui retournera `true` si le joueur est arrivé à la case finale, et une méthode `toString()` qui indiquera qui il est, à quelle position il est arrivé et s'il a gagné.

Complétez votre classe avec les méthodes et constructeur(s) qui vous semblent nécessaires.

Faire une classe `lancerJeu` qui lancera le jeu

Un affichage basique peut être :

```
Combien de joueurs voulez-vous ?
3
joueur num 1 a la position 0
joueur num 2 a la position 0
joueur num 3 a la position 0
appuyez sur enter pour continuer

joueur num 1 a la position 2
joueur num 2 a la position 5
joueur num 3 a la position 3
appuyez sur enter pour continuer

.....
```

Exercice 5 Pour aller plus loin (facultatif)

Essayez maintenant d'adapter votre programme au jeu de l'oie, donc les règles sont à la page

`http://fr.wikipedia.org/wiki/Jeu_de_l'oie` .

Vous pouvez bien sûr ne pas implémenter toutes les règles ou en inventer d'autres.