Université Paris Diderot **Programmation Orientée Objet**
L2-L3 **Année 2014-2015**

TP n° 6

Classes internes

Exercice 1 (Boxing, unboxing)
Dans une classe `Main`, écrivez une méthode statique `int f(int x)` qui retourne son argument. Idem pour la méthode `Integer g(Integer x)`.

Dans le `main`, essayez d'appeler la méthode `f` avec comme argument un `Integer` et la méthode `g` avec comme argument un `int`. Que se passe-t'il ?

Peut-on écrire `Integer nb = 1;` ?

Peut-on écrire `int n = new Integer(1);` ?

Testez.

Attention ! Il ne faut abuser de ce phénomène.

Exercice 2 (classes internes statiques, méthode clone)
Créez une classe `HorlogeDigitale` qui permettra l'affichage d'une heure sous le format "HH :MM", par exemple "21 :30".

Dans cette classe, vous créerez une classe interne statique `Digit` qui contiendra deux entiers, un pour pour la valeur du chiffre, et un autre pour la valeur maximum du chiffre. Faites un constructeur qui prend seulement la valeur maximum en argument et un avec deux arguments. Dans `Digit` il y aura une méthode `getValeur()`, une méthode `incremente()`, une méthode `estALaValeurMax()` et la redéfinition de `toString()`. La méthode `incremente` ajoutera 1 à la valeur, jusqu'à ce que la valeur maximum soit atteinte, puis reviendra à 0.

Maintenant, vous pouvez définir la variable d'objet de `HorlogeDigitale`, qui sera une tableau de 4 digits. Ajoutez les constructeurs nécessaires et une méthode `incremente()`, redéfinissez `toString()`. Il faudra faire attention au cas où l'on passe de "23 :59" à "00 :00".

Implémentez aussi une méthode `clone()` (cf. cours) pour `HorlogeDigitale` de telle manière que l'horloge clonée devra être indépendante de l'horloge originelle.

Exercice 3 (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.

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 et à quel position il est arrivé, il indiquera aussi 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

.....
```

Pour aller plus loin 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.

Exercice 4 (Exercice supplémentaire : méthode finalize)

On veut tester la méthode `finalize()`.

Pour ce faire, créez une classe `Truc` dont les objets occupent une grosse place en mémoire (vous avez le choix), ces objets auront un numéro. Le mieux est de faire des objets dont la taille est paramétrable.

Le constructeur, en plus de créer l'objet écrira quelque chose du genre "creation du Truc numero 6". Implantez aussi la méthode `finalize()`, elle écrira quelque chose du genre "destruction du Truc numero 6".

Maintenant faites une boucle qui va créer beaucoup de `Truc` (sans garder leur adresse en mémoire), si vous ne voyez jamais apparaître "destruction du Truc ..." c'est que vous n'avez pas créé assez de `Trucs` et/ou qu'ils ne sont pas assez gros.

Pour mieux voir ce qui se passe vous pouvez utiliser

```
try{
  Thread.sleep(10);
} catch(InterruptedException e){}
```

qui va attendre 10 millisecondes avant de reprendre, vous pouvez bien sûr changer la valeur en paramètre de `sleep`.