TDLOG – séance n° 3
Python orienté objet : TP

Xavier Clerc – xavier.clerc@enseignants.enpc.fr
Thierry Martinez – thierry.martinez@inria.fr
Olivier Cros – olivier.cros0@gmail.com

10 octobre 2016

À rendre au plus tard le 15 octobre 2016 sur educnet

On se propose au cours des séances qui viennent de concevoir un jeu simple à deux joueurs. L'objectif pour aujourd'hui est de décrire les règles du jeu en *Python* (le *modèle* du jeu) et de concevoir une interface texte rudimentaire pour y jouer. En utilisant le paradigme objet, nous veillerons à ce que notre programme soit extensible de façon à pouvoir écrire des interfaces graphiques plus sophistiquées lors des séances à venir.

1 Règles du jeu

Sur une grille sont positionnés un personnage ainsi que des pièces de différentes valeurs (5, 10, 20, 50, 100 et 200). La grille est carrée et de côté impair. Le personnage est initialement placé sur la case centrale de la grille. Exceptée la case du personnage, toutes les cases contiennent une pièce dont la valeur est aléatoire.

Les deux joueurs jouent chacun leur tour. À chaque tour, le joueur choisit une direction dans laquelle déplacer le personnage, d'une seule case : sur la gauche, sur la droite, en haut, en bas, ou en diagonale, avec les deux contraintes qu'il ne faut pas que le personnage sorte de la grille, et que la case sur laquelle le personnage arrive doit contenir une pièce, que le personnage ramasse au passage.

Si un joueur ne peut plus déplacer le personnage selon ces contraintes, la partie prend fin et le vainqueur est le joueur dont les pièces rassemblent la plus grande valeur.

2 Implémentation du *modèle*

2.1 Grille

Écrire une classe `Grid` implémentant la grille, dont le constructeur est paramétré par une taille. La classe maintiendra dans son état interne une grille du format demandé, et permettra de lire la taille, ainsi que de lire et d'écrire individuellement dans chaque cellule de la grille. On ne spécifie pas à ce niveau comment le contenu de chaque cellule est représentée : il peut s'agir de n'importe quelle valeur du langage. On initialisera simplement les cellules à la valeur `None`.

On surchargera l'opérateur `in` de façon à ce qu'on puisse écrire `(x, y) in grid` pour savoir si `(x, y)` sont des coordonnées valides pour le format de `grid`.

2.2 Joueur

Écrire une classe `Player` pour représenter un joueur, caractérisé par son nom et son score. Le constructeur permettra de passer un nom, celui-ci sera ensuite accessible en lecture seule. Le score sera initialisé à 0 : on pourra le lire, et une méthode permettra de l'incrémenter.

2.3 État de la partie

Écrire une classe `Game` dont le constructeur sera paramétré par une taille et des noms de joueurs. La classe maintiendra une liste des joueurs dans son état interne, instances de la classe `Player`, ainsi qu'une grille de la taille passée et l'identité du joueur courant.

Augmenter la classe de méthodes permettant de :
- jouer un coup pour le joueur courant ;
- savoir si la partie est terminée ;
- déterminer le résultat de la partie, si elle est terminée.

3 Implémentation du jeu en mode texte

3.1 Affichage

Ajouter à `Grid`, `Player`, et `Game` les méthodes nécessaires à l'affichage à l'écran de la grille et des scores. Par exemple, un état initial du jeu peut être affiché comme suit :

```
 20  100   10   50    5
100  200   10  100  200
200  100  ###    5   50
200  200  200  200  100
 50    5   50   10  200

N1: 0
```

```
N2: 0
```

où `###` représente le personnage, et `N1` et `N2` les noms des joueurs.

3.2 Interaction

Ajouter les méthodes nécessaires permettant aux joueurs d'entrer à tour de rôle leurs déplacements, en mettant à jour puis affichant la grille à chaque coup. La fonction `input` peut être utilisée pour lire une chaîne de caractères au clavier ; elle accepte un paramètre optionnel utilisé comme *prompt*. La chaîne, encodant le movement, peut par exemple être `N` pour le nord, `S` pour le sud, *etc.* Lorsque la partie est terminée, il faut afficher le nom du vainqueur s'il y en a un, ou que la partie est un match nul.