Introduction

Ce rapport à pour but de présenter le projet que j’ai réalisé dans le cadre du cours de Java, lors de ma 4ème année d’étude à Polytech Clermont.

Ce projet a consisté en l’implémentation du jeu « Puissance 4 », à l’aide de la programmation orientée objet.

Pour ce faire, j’ai modélisé plusieurs classes Java, permettant de modéliser des entités du jeu, telles que la grille du puissance 4, les cases de la grille, les coordonnées des cases, etc.

Nb : Afin de versionner ce projet, de le partager, ainsi que d’obtenir une sauvegarder Cloud de ce dernier, j’ai créé un repository GitHub pour ce programme : <https://github.com/Nathan7128/projet-puissance4>.

Annonce du plan

Dans un premier temps, je vais présenter brièvement le principe du jeu « Puissance 4 ».

Dans une seconde partie, j’expliquerai le principe de fonctionnement du programme, en expliquant la logique utilisée pour modéliser le jeu à l’aide de la POO.

Ensuite, je présenterai plus en détail les différentes classes Java du programme, en donnant des précisions sur leurs attributs et méthodes.

Enfin, j’expliquerai les difficultés que j’ai pu rencontrer durant l’implémentation de ce code.

Présentation du jeu

Le puissance 4 est un jeu composé d’une grille, qui nécessite 2 joueurs. Chaque joueur peut glisser un « jeton » dans une colonne, et gagne lorsqu’il a aligné le nombre de jetons requis pour gagner.

L’alignement peut être horizontal, vertical ou diagonal.

Dans le jeu que j’ai codé, il est possible de choisir la largeur de la grille (le nombre de colonnes), la hauteur de la grille (le nombre de lignes), ainsi que le nombre de pions à aligner pour gagner la partie.

Fonctionnement du programme

Le puissance 4 est modélisé par une grille (cf. image 1) (avec une largeur et une hauteur), qui est implémentée par un « HashMap » Java.

Ce HashMap est en réalité un dictionnaire clé/valeur, qui possède comme clé des objets de type Coordonnées (qui ont comme attributs une ligne et une colonne), et comme valeur des cases de la grille.

Une autre solution pour implémentée la grille aurait été de l’implémentée à l’aide d’une matrice de case (ou chaque élément de la matrice correspond à une case de la grille), cependant, il nous a été demandé d’utiliser ce HashMap.

Il est donc possible de retrouver une case de la grille grâce à un objet de type Coordonnées. Chacune de ces coordonnées possédant un « Hash code », ainsi qu’une méthode permettant de déterminé si 2 coordonnées sont égales.

J’expliquerais plus en détail dans la section sur la classe Coordonnées le fonctionnement de cet hash code et de la manière de comparer 2 objets de type Coordonnées.

Pour les cases, elles peuvent être de 3 types : vides, jaunes ou rouges.

Les cases vides représentant des cases dans lesquelles les joueurs n’ont pas placé de pion, les cases jaunes représentent les cases dans lesquelles le joueur numéro 1 à jouer, et les cases rouges celles du joueur numéro 2.

Ces 3 types sont implémentés comme des classes héritées de la classe mère « Case », qui permet de définir les points communs de ses classes filles.

En ce qui concerne la partie, il est permis aux joueurs de choisir la largeur et la hauteur de la grille, ainsi que le nombre de pions à aligner pour gagner.

La partie est jouée par 2 joueurs, qui sont identifiés comme « Joueur 1 » et « Joueur 2 ».

Le joueur 1 joue avec les cases jaunes, et le joueur 2 avec les cases rouges.

Tant que la partie n’est pas terminée (tant qu’aucun joueur n’a gagné la partie et que la grille n’est pas pleine), chaque joueur joue à son tour un pion dans la grille.

A chaque coup d’un joueur, on ajoute au HashMap le couple (« coordonnées de la case jouée », « objet de type case rouge/jaune » (en fonction du joueur qui a effectué le coup) ).

Pour déterminer si une partie est finie, on regarde après chaque coup d’un joueur si la grille est pleine ou si le joueur qui a joué à aligner le nombre de pions requis pour gagner. Pour vérifier cette deuxième condition, on regarde pour chaque case (de la couleur correspondant au joueur qui a joué), 4 alignements possibles :

* La ligne droite : le nombre de cases (de la bonne couleur) alignées horizontalement avec la case qu’on regarde.
* L’alignement bas : le nombre de cases (de la bonne couleur) alignées verticalement avec la case qu’on regarde.
* La diagonale en bas à droite : le nombre de cases (de la bonne couleur) alignées dans la diagonale en bas à droite de la case qu’on regarde.
* La diagonale en bas à gauche : le nombre de cases (de la bonne couleur) alignées dans la diagonale en bas à gauche de la case qu’on regarde.

En itérant ces différentes cases, en commençant par celle en haut à gauche de la grille et en se déplaçant de gauche à droite puis de bas en haut, il est possible de vérifier l’ensemble des alignements possibles pour l’ensemble des cases.

Lorsque la partie est finie, les joueurs choisissent ou non de recommencer une nouvelle partie.

Dans le cas ou ils en recommencent une, il est donc une nouvelle fois demandé à l’utilisateur la largeur et la hauteur de la grille, ainsi que le nombre de pions à aligner.

Puis, une nouvelle partie est instanciée, avec les choix effectués par l’utilisateur.

1. La classe Coordonnées

Comme son nom l’indique, cette classe permet de modéliser les coordonnées des cases de la grille.

Ces coordonnées sont exprimées en ligne (attribut « i » de la classe) et en colonne (attribut « j »).

Les coordonnées i = 0 et j = 0 correspondent à la case située en haut à gauche de la grille.

Comme expliqué dans l’explication du programme, ces objets de type Cordoonées sont utilisés comme clé d’un HashMap. De ce fait, il a fallu redéfinir la méthode permettant d’obtenir un hash code d’un objet de type Coordonnées, ainsi que la méthode permettant de savoir si deux coordonnées sont égales.

En ef