RAPPORT DE PROJET JAVA

--

Clone du Jeu Puissance 4

Étudiant : SOGLO Arcadius

Enseignant : Damien LOLIVE

IMR 2018

SOMMAIRE

Introduction

Pour mettre en pratique nos connaissances acquises en cours, nous sommes amenés à réaliser un projet en monôme. L'objectif est de réaliser un clone du jeu puissance 4 en utilisant le modèle MVC (Modèle Vue Contrôleur) et la librairie Swing.
Le cahier des charges est de réaliser le clone du jeu en respectant certaines contraintes :

-taille du plateau configurable
-interaction avec l'utilisateur : saisie des noms/pseudos des joueurs
-affichage des meilleurs scores (et sauvegarde dans un fichier).

- Les outils utilisés pour mettre en œuvre le clone du jeu puissance 4 sont :
- JAVA : Langage de programmation orienté objet.
- JDK 1.8.0
- Librairie Swing
- Java2D
- MVC
- IDE Eclipse

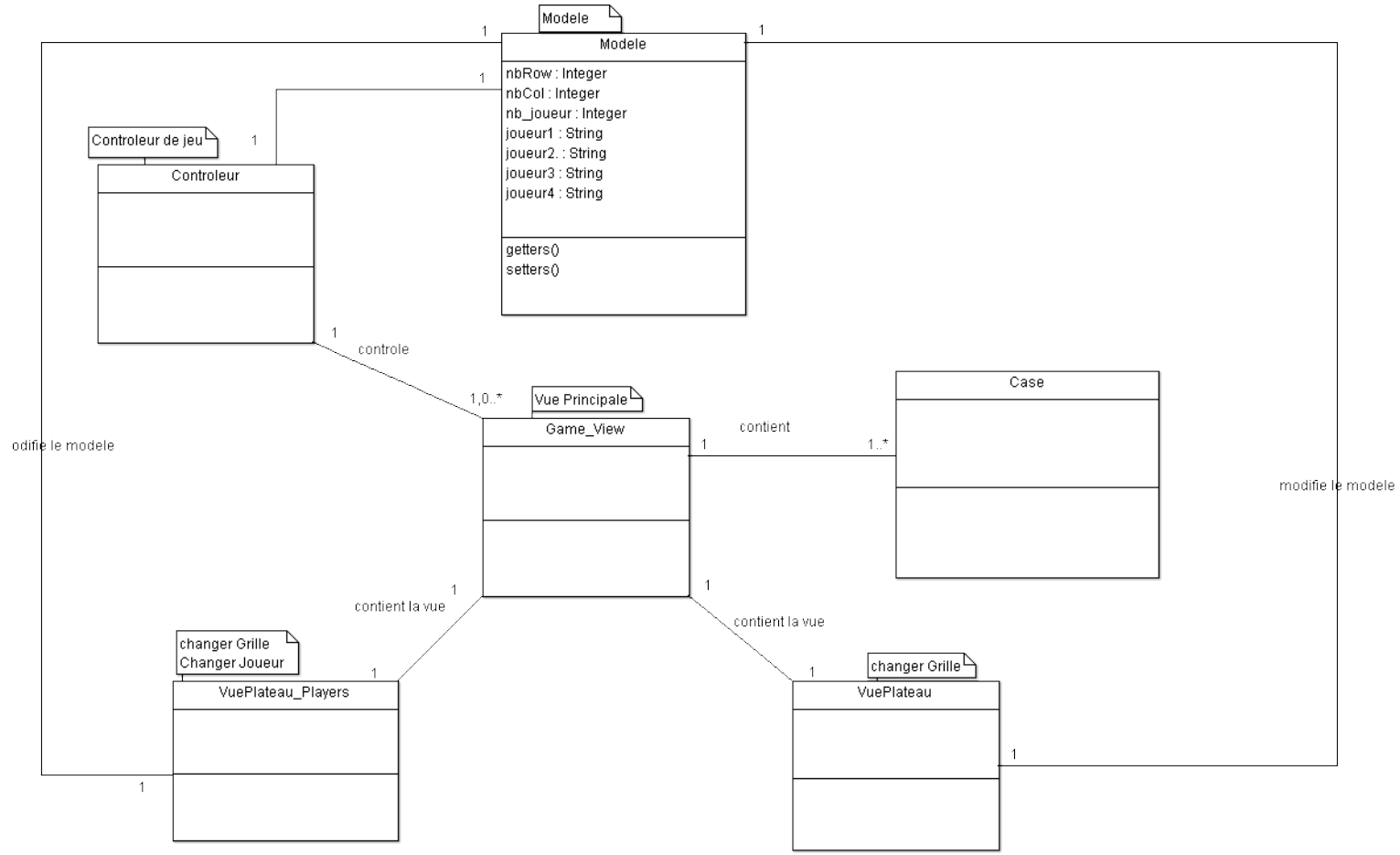
Après avoir analysé le cahier des cahiers et proposer un modèle pour répondre à celui-ci, nous analyserons les résultats obtenues et les choix effectués.

I. Analyse du cahier des charges

Après analyse du cahier des charges et du fonctionnement du jeu puissance 4, voici le diagramme de classe qui illustre ma solution pour le clone du jeu.

Ce diagramme omet une classe : la classe PopUp, qui est une classe qui me permet d'afficher des messages (erreur, ou informations). Sa présence sur le diagramme n'est pas importante car elle n'influence pas le fonctionnement normal du jeu. La classe PopUp figure dans le fichier source rendue avec ce rapport.

1. Diagramme de Classe : UML



2. Explication du diagramme et des choix effectués.

Le Modèle : La classe Modèle

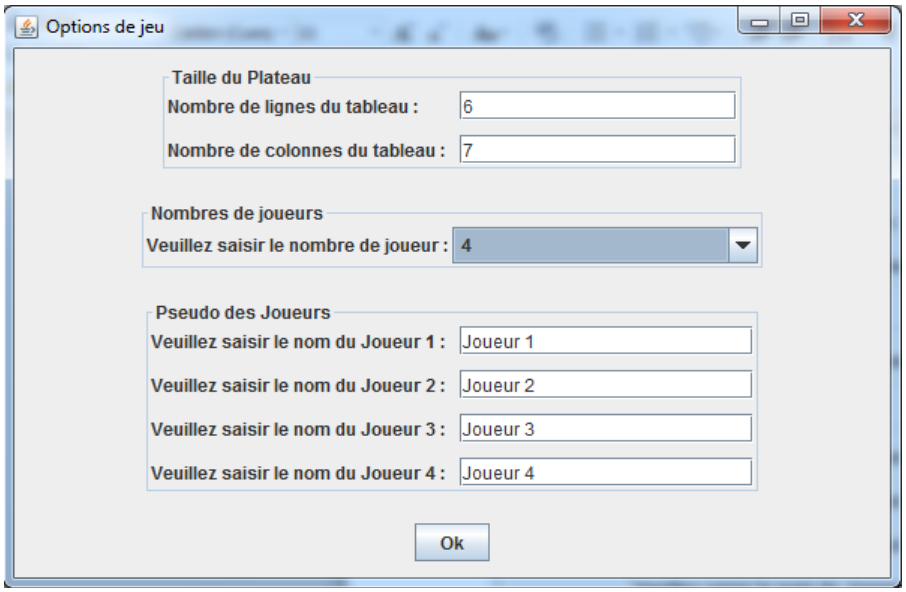
Il s'agit du modèle de notre Jeu de puissance 4. Elle définit les caractéristiques du jeu : nombre de joueur, nombres de lignes, nombres de colonnes, nom des joueurs. Elle fournit des méthodes pour modifier ou récupérer ses valeurs. Il s'agit du cœur de notre application.

Elle contient et gère les différentes options de jeu (dimensions du jeu, nom des joueurs...).

Une fois le modèle créé, il nous faut des vues pour interagir avec celui-ci. La classe VuePlateau_Players est une interface Swing qui permet de modifier les caractéristiques de notre modèle. J'utilise le **Pattern Observable/Observer** pour notifier au modèle les changements et mettre à jour la vue constamment.

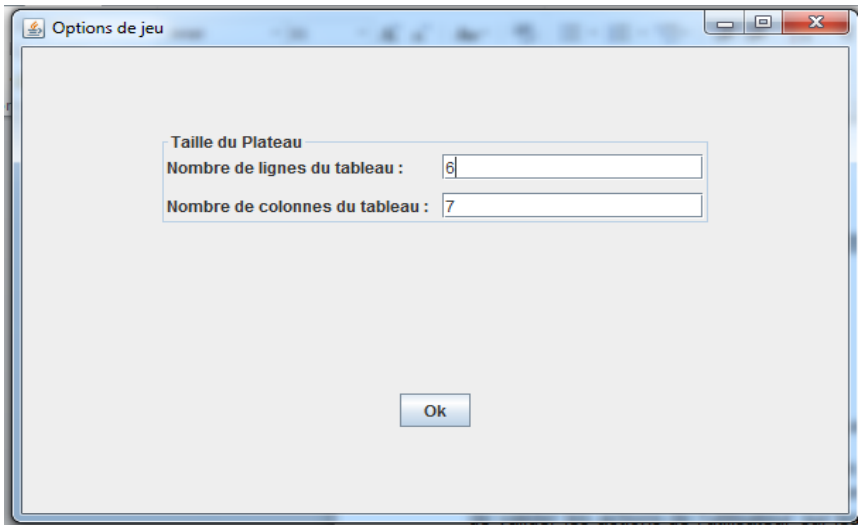
La vue : VuePlateau_Players:

Elle définit de façon graphique la fenêtre d'options (hérite donc de **JFrame**) et transmettra les paramètres entrés au Modèle. Il s'agit de l'interface graphique complète pour mettre à jour toutes les caractéristiques du modèle.



La vue : VuePlateau :

Elle définit de façon graphique une autre fenêtre d'options (hérite aussi de **JFrame**) et transmettra les paramètres entrés au Modèle. La différence avec VuePlateau_Players est que cette interface permet uniquement la mise à jour de la dimension du plateau de jeu. Le fonctionnement reste le même que VuePlateau_Players (Observable/Observer). Cette interface est uniquement utile pour permettre aux utilisateurs de changer la dimension du plateau de jeu avant le début de la partie. Un bouton sur la vue principale permettra aux joueurs de modifier leur choix de plateau.



Case :

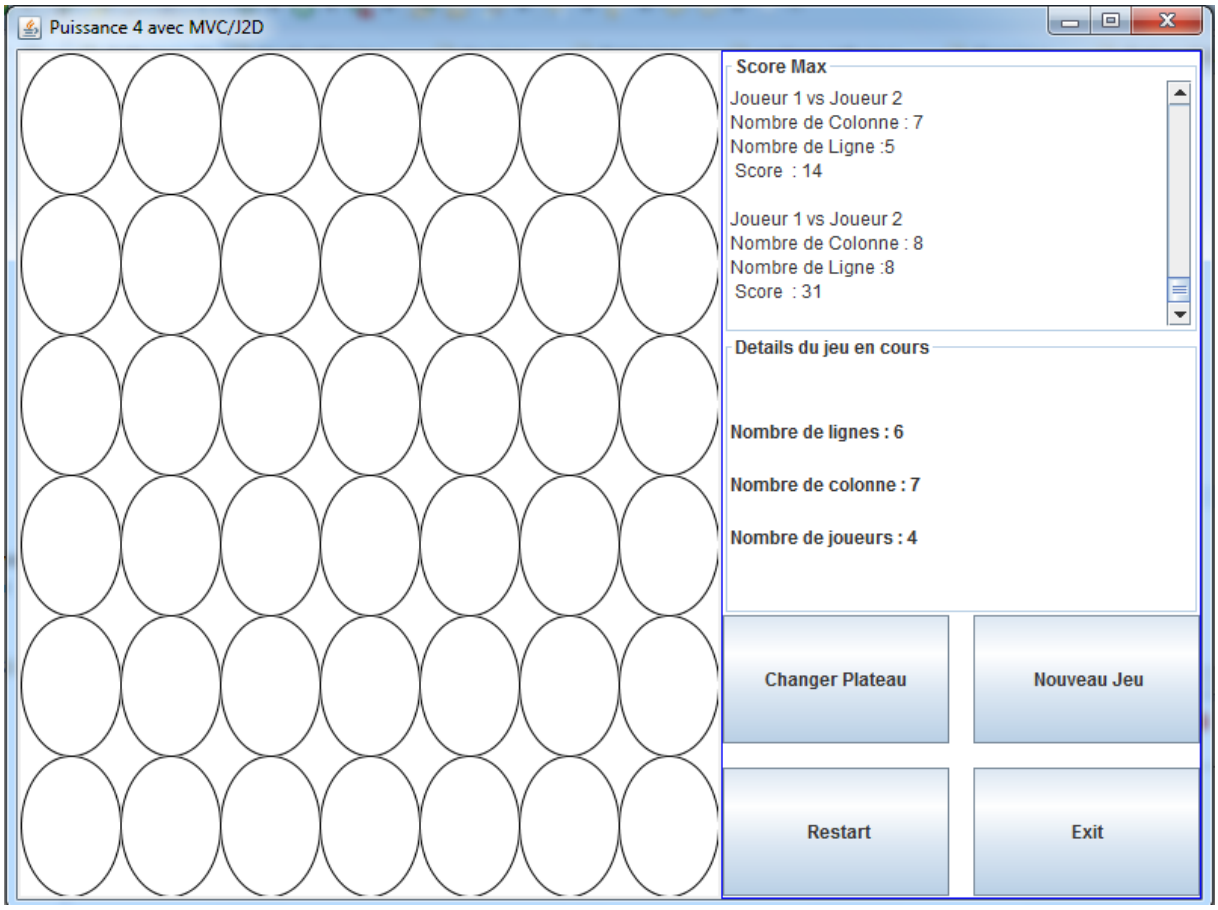
La classe case permet la gestion graphique d'une seule case, un plateau de jeu étant composé de plusieurs objets de type Case(**JPanel**). En fonction du nombre de ligne et de colonne saisie par l'utilisateur, je dessine un nombre de case sur le plateau de jeu. Une méthode récupère du modèle le nombre de ligne et de colonne puis avec un GridLayout on dessine le plateau de jeu.

Ayant la volonté de gérer le plateau de jeu sous forme de x cases sur lequel je place des écouteurs pour contrôler le positionnement de pions. Un contrôleur de jeu me permettra de valider les actions de l'utilisateur sur les différentes cases du plateau.

La vue principale : Game_View

Elle définit la fenêtre de jeu (barre de statut, plateau de jeu (nombres de cases) et la gestion de celles-ci par la souris, panneau latérale avec informations sur la partie en cours). Elle étend donc **JFrame**. C'est notre **Vue principale** qui contient l'ensemble des éléments graphiques pour interagir avec notre jeu. Elle contient donc un ensemble de cases gérer avec la souris (Plateau de jeu). Les vues Vue Plateau_Players et Vue Plateau sont contenues dans cette fenêtre principale. Elle sont appelles avec les boutons Changer Plateau et Nouveau Jeu. Les scores sont chargés dans la barre latérale de cette vue à l'ouverture du jeu grâce au fichier scoremax.txt.

Voici l'interface graphique principale du jeu :



Le contrôleur de Jeu :

C'est le contrôleur de jeu et aussi la classe principale appelée. Elle fournit des méthodes pour connaître le gagnant, valider un coup, ouverture d'une partie de jeu. Elle fait appel à la vue Game_View. Ayant considéré dans un premier temps le jeu pour 2 joueurs, j'ai mis en place les méthodes pour savoir si un joueur à gagner ou pas vérifiant sur l'horizontal, la vertical et les 2 diagonales. Ayant des mauvais résultats lors des tests pour 4 joueurs, le code source fournie prend en compte que le fonctionnement avec 2 joueurs.

Cependant les vues permettant de faire les choix du nombre de joueur sont présent sur l'interface mais seul 2 joueurs seront pris en compte lors du jeu.

La classe Controleur est celle a instancié car elle permet de gérer les différentes actions sur le plateau de jeu contenu dans la vue principale et elle permet de gérer la matrice de jeu et le calcul des gagnants et la validation des coups entre autre.

II. Fonctionnement du jeu

PUISSANCE4 est un jeu à deux joueurs bien connu. Le but du jeu est d'aligner quatre jetons avant l'adversaire

- Les joueurs jouent à tour de rôle et ne peuvent pas passer leur tour
- Le jeu utilise une grille verticale composée de 7 colonnes et 6 lignes par défaut
- Une partie commence avec une grille vide. Un joueur dispose de jetons d'une même couleur
- Un joueur rempli la grille en laissant «tomber» un jeton au sommet d'une colonne de son choix
- Un joueur a gagné dès qu'il a aligné (verticalement, horizontalement ou bien en diagonal) quatre jetons
- Une partie est déclarée nulle quand il n'y a pas d'alignement de 4 jetons et que toutes les colonnes sont remplies

Au lancement, de la partie, la fenêtre (Vue_Plateau_Players) invite les joueurs à saisir leur pseudo et le nombre de ligne et de colonne pour la partie. Je n'ai pas pu rendre dynamique le choix du nombre de joueurs et la saisie des pseudos, c'est à dire en fonction du nombre de joueurs choisit afficher 2,3 ou 4 zones de saisie pour les pseudos.

Une fois les caractéristiques saisies, cette fenêtre se ferme, et la vue principale s'affiche. Cette vue principale(Game_View) contient une synthèse sur le panneau latéral des choix effectués. Sur le panneau de gauche, le plateau de jeu est dessiné sous forme de x cases. Une barre de statut indique l'utilisateur qui joue sa partie. Des fenêtres d'interactions s'affichent pour informer le joueur en cas de mauvais coup (colonne remplie par exemple).

III. Difficultés rencontrées et avancement du travail

Au début de mon développement, j'ai considéré, qu'on dispose au minimum de 2 joueurs pour jouer la partie. Pour connaître le gagnant d'une partie j'utilise le tableau à 2 dimensions dans lequel je place des numéros associé à chaque joueur. Je me suis fixé comme objectif de faire fonctionner le jeu pour deux joueurs ensuite l'améliorer pour 4 joueurs. Malheureusement, je n'ai pas réussi à le faire fonctionner pour quatre joueurs par manque de temps car des erreurs se produisent. Le jeu néanmoins détecte le gagnant peu importe la ligne, la colonne et les diagonales pour 2 joueurs.

IV. Tests et analyse des résultats.

Voir vidéo de démonstration

V. Fonctionnalités supplémentaires :

Les attentes du cahier de charges sont respectées. J'ai voulu rendre le jeu interactif avec l'utilisateur alors j'ai utilisé les méthodes de la classe WindowListener pour afficher un message d'accueil et de fin. De même la gestion des actions de l'utilisateur est très interactifs (messages d'erreur en cas de mauvaise saisies de valeurs pour l'initialisation du plateau de jeu). La classe PopUp me permet d'afficher des messages en cas de mauvaises opérations par l'utilisateur (colonne jouée pleine, valeur de la dimension du plateau saisie incorrect,…)

Conclusion

Ce projet m'a permis de mettre en œuvre mes connaissances acquises durant les séances de cours et de TP. J'ai pu approfondir mes connaissances sur la librairie Swing et le positionnement des éléments avec Swing. Malgré les difficultés rencontrées pour réaliser un jeu fonctionnel, j'ai pu réaliser un clone du jeu puissance 4 fonctionnelle pour 2 joueurs. En pièces-jointe se trouve le fichier source de mon application avec une vidéo du fonctionnement. Le fichier soglo.jar est le script de compilation pour jouer au jeu puissance4. (java –jar soglo.jar).