IUT du Havre
Université du Havre
Année Universitaire : 2015-2016

Manuel du programmeur

Programme SolitaireChess

Boulant Florian
Di Gregorio Thomas
Edouard Clémence
Emion Thibaut

Présentation

Bienvenue dans ce manuel dédié au développeur qui reprendra notre projet de Solitaire Chess. Nous allons dans ce manuel vous présenter notre conception du programme, nous listerons les fonctionnalités principales de celui-ci, et enfin nous détaillerons les plus techniques d'entre elles.

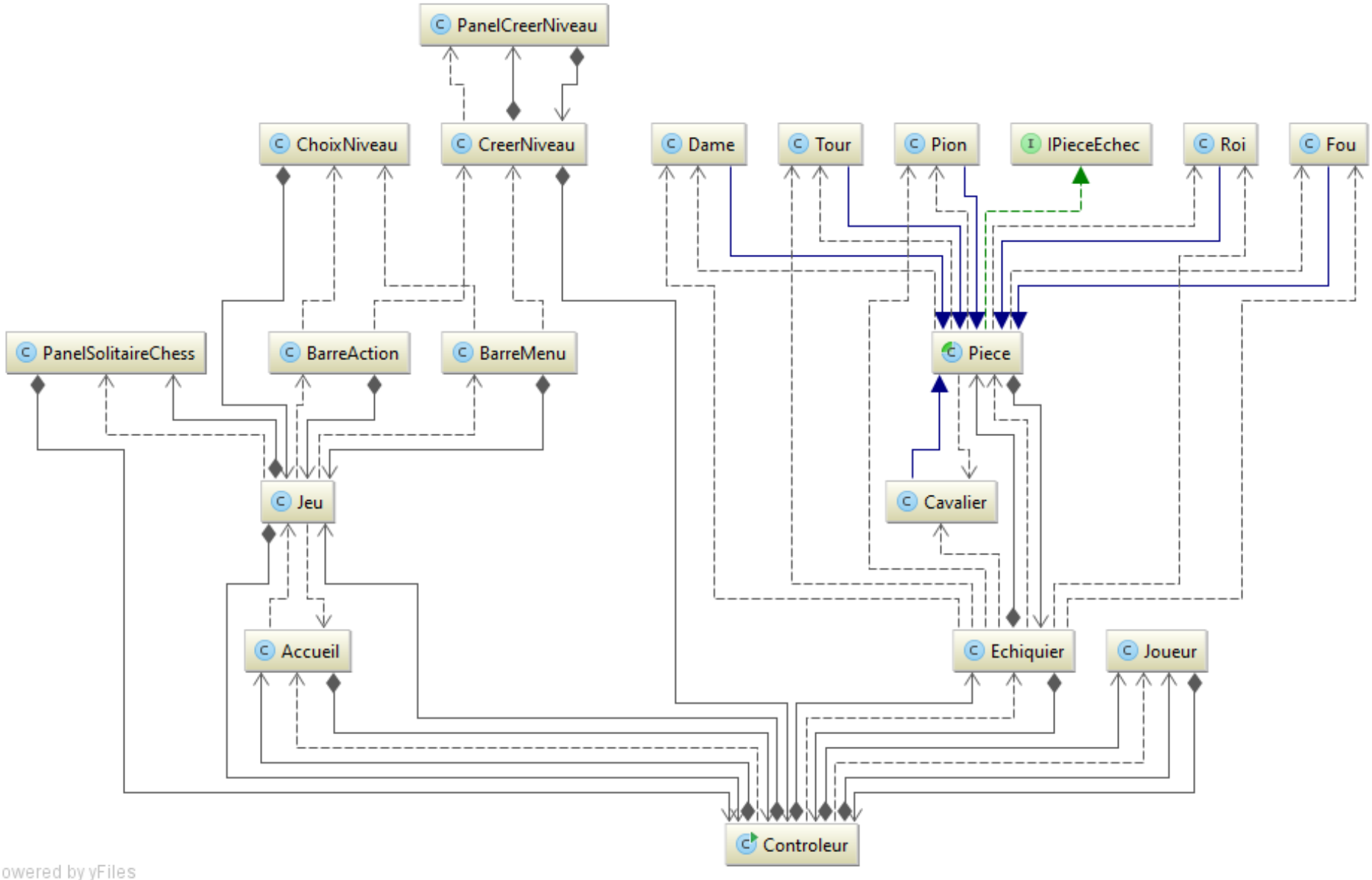
La conception du programme

Nous allons dans cette partie vous présenter sous forme de diagramme UML notre conception du programme, puis nous l'expliquerons afin que vous compreniez bien les règles que l'on a établies.

Le diagramme

Vous retrouverez le diagramme à la page suivante.

Vous pouvez voir qu'on a séparé les interfaces graphiques de la gestion « métier » de l'échiquier. La séparation est nette et doit le rester. Nous souhaitons que la classe *Controleur* soit la classe qui fera l'intermédiaire entre les paquetages *métier* et *ihm*. On peut observer que la classe *Piece* est abstraite (un quart du cercle vert sur le diagramme selon les conventions d'IntelliJ), et que les pièces héritent toutes de cette classe. L'interface *IPieceEchec* est utilisée dans le programme afin que l'IHM interagisse avec le métier sans lien direct. Cette interface est utilisée dans le programme pour la surbrillance des cases vers lesquelles on pourrait aller.


PanelCreerNiveau
ChoixNiveau
CreerNiveau
Dame
Tour
Pion
IPieceEchec
Roi
Fou
PanelSolitaireChess
BarreAction
BarreMenu
Piece
Jeu
Cavalier
Accueil
Echiquier
Joueur
Controleur
Powered by yFiles

Les règles de programmation

Elles sont :

- Noms des attributs et méthodes en français.
- Garder la même syntaxe pour toute la Javadoc.
- Respecter le style de code utilvisé dans les classes.
- Il faudra commenter chaque partie technique.

Nous tenons à ce que vous respectiez ces règles, afin de garder une cohérence dans les potentiels ajouts que vous feriez.

Liste des fonctionnalités

Nous dresserons dans cette partie une liste qui se voudra la plus exhaustive possible. Veuillez la compléter s'il vous arrive d'ajouter de nouvelles fonctionnalités.

Fonctionnalités :

- Le déplacement des pièces.
- La possibilité de recommencer un défi (et non niveau, parce que les niveaux représentent les difficultés).
- La possibilité d'annuler un déplacement.
- La possibilité d'avoir des indices.
- La possibilité de sauvegarder sa partie dans un profil.
- La multiplicité des sauvegardes.
- La gestion des sauvegardes (supression, ajout).
- La surbrillance des cases vers lesquelles on peut aller.
- Le menu de choix de défi.
- Le menu de création de défis.
- Les thèmes que l'on peut changer, et ajouter (assez facilement).
- Le déblocage progressif des niveaux *classiques* (dans le jeu de base).

Détail de certaines fonctionnalités

Déplacer

Cette fonctionnalité est un bon exemple pour montrer la séparation nette entre *ihm* et *métier*. Deux méthodes sont à l'origine de l'appel de la méthode *deplacer*, ce sont des méthodes de gestion d'événements, les méthodes *mouseClicked* et *mouseReleased*. Vous pourrez observer que dans déplacer on vérifie si la pièce peut se déplacer selon son paterne aux échecs, puis si elle n'enjambera pas de pièce dans le processus, ceci est vérifié avec *peutSeDeplacer* (booléen) et *personneDansLeChamp* (booléen).

Annulation

L'annulation est certainement l'une des fonctionnalités les plus techniques du programme. On stocke dans *Echiquier* une *ArrayList* de *Tableaux de Piece* qui contiendra l'état du plateau avant et après chaque déplacement. Quand l'utilisateur cliquera sur le bouton prévu à l'annulation de son coup, on restituera l'état précédent du plateau dans le plateau actuel de l'*Echiquier*.

Les thèmes

Les thèmes permettent de changer l'affichage du jeu. Ils sont modifiés pour chaque joueur dans la classe *Joueur*, et définis dans la classe *PanelSolitaireChess*.

```
private static final Color[][][] TAB_THEMES = new Color[][][]{ { { new Color( 0x34B618 ),
                                                                   new Color( 0xFFDF50 ),
                                                                   new Color( 0xBF1C02 ),
                                                                   new Color( 0x9C000F ) },
                                                                 { new Color( 0xF0FFEB ) } },

                                                               { { new Color( 0x4BFF9D ),
                                                                   new Color( 0x3D79FF ),
                                                                   new Color( 0xBC9EFF ),
                                                                   new Color( 255, 0, 250 ) },
                                                                 { new Color( 0x0D0418 ) } },

                                                               { { new Color( 0xFFB8EE ),
                                                                   new Color( 0xFD87FF ),
                                                                   new Color( 0xFC50FF ),
                                                                   new Color( 248, 0, 255 ) },
                                                                 { new Color( 0x000000 ) } } };
```

Pour chaque thème, on peut définir de nouvelles images, et de nouvelles couleurs. Les images seront définies dans le dossier *images* dans un dossier intitulé *thème%02d* où on mettra à la place de %02d le numéro du thème sur deux chiffres. On devra aussi ajouter un thème dans le tableau de *PanelSolitaireChess* en faisant comme sur l'exemple ci-dessus.

La multiplicité des sauvegardes

Les sauvegardes sont faites dans un fichier *sauvegarde.data* à la racine du programme. On utilise le principe de sérialisation pour garder les données sur les joueurs, joueurs qui sont stockés dans une *ArrayList* nommée *alJoueurs*.

lol