Rapport

April 10, 2020

Ce projet a pour but de réaliser une application simulant un jeu de taquin ainsi qu'une interface graphique permettant de jouer à ce jeu. La réalisation de cette application a été intégralement faite selon une architecture MVC de manière a ce que le modèle soit indépendant de l'interface graphique. Notre code contient donc trois packages, un pour le modèle, un pour la vue et un pour le contrôleur.

Contents

1 Architecture du code

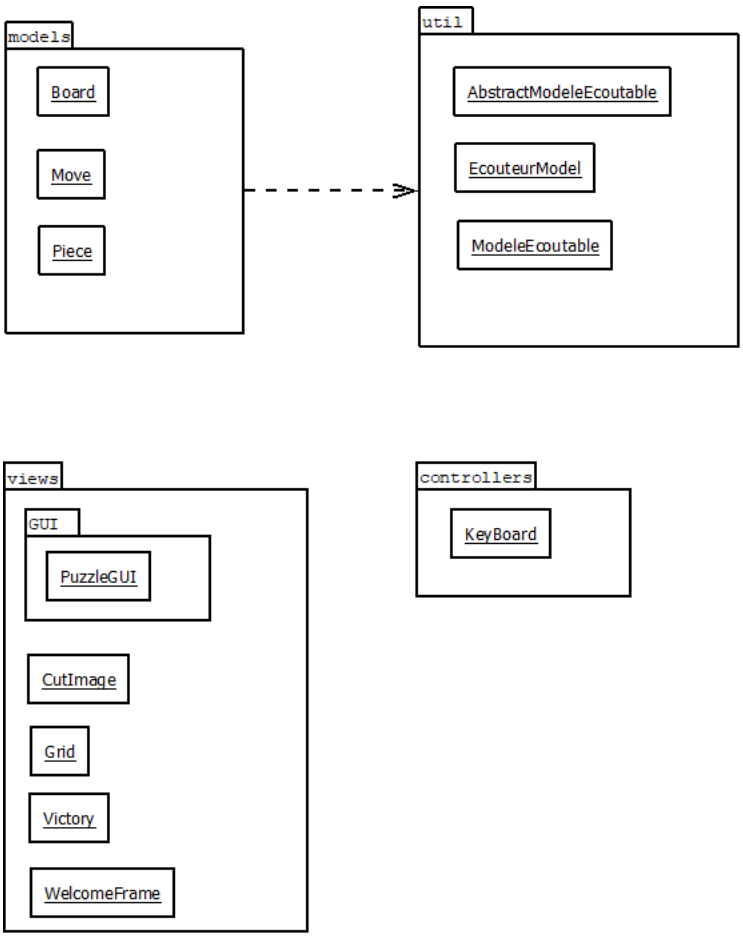

Figure 1: *D*iagramme des packages

Le package plateau contient le code qui permet la modélisation de notre jeu et des éléments qui le constitue. Ce package contient trois classes. La classe Board qui représente notre plateau de jeu, la classe Piece qui permet de caractériser les différentes pièces de notre plateau, et la classe Move qui permet de représenter un mouvement de pièces possible dans notre jeu.

2 Eléments techniques

2.1 Fonctionnement du modèle

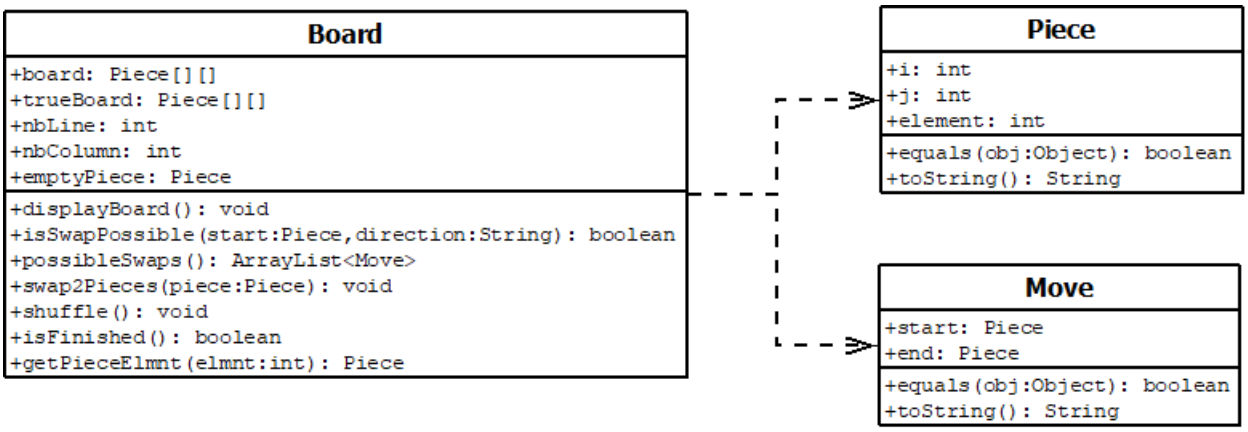

Figure 2: *D*iagramme du package models

Nous avons fait le choix que dans notre classe Board, le plateau soit représenté par une matrice d'éléments de type Piece.La classe Piece est définie par ses coordonnées sur le plateau et un élément qui représente le contenu de cette pièce. Dans notre code les coordonnées des pièces ne varient jamais, lorsqu'une pièce bougera ce sera son élément de contenu qui sera modifié.

Le plateau de jeu est construit dans la classe en Board en créant la matrice d'éléments de type Piece, d'une taille qui est définie par deux attributs (nbLine et nbColumn) ce qui va nous permettre par la suite de de faire choisir au joueur la taille du plateau de jeu. Lors de la construction du plateau de jeu celui-ci est parcouru pour initialiser le contenu de toutes les pièces dans l'ordre. De plus une copie de ce plateau est faite dans un attribut appelé trueBoard. Ensuite, toujours durant la construction, les pièces du plateau sont mélangées grâce à la fonction shuffle qui pour cela joue un grand nombre de coups aléatoires.

Un mouvement est caractérisé par deux attributs dans notre classe Move : un pour la pièce de départ et un autre pour la pièce d'arrivée La fonction isSwapPossible permet de regarder si une pièce est collé à l'une des quatre

bordures du plateau en fonction de la direction donnée en paramètre et ainsi de savoir s'il sera possible de déplacer cette pièce dans cette direction. Pour connaitre les mouvements de pièces qui peuvent être réalisés, nous utilisons la méthode possibleSwaps qui récupère la position de la pièce vide est qui teste, grâce à la fonction isSwapPossible, dans combien de position elle peut être déplacée. Les résultats trouvés sont placés sous forme de Move dans une liste renvoyée par la méthode.

La fonction permettant de réaliser un déplacement de pièces est appelé swap2pieces et prend en paramètre la pièce que l'on veut déplacer sur la case vide. Cette fonction regarde si le mouvement entre la pièce vide et celle donnée en paramètres appartient bien à la liste des mouvements possibles. Si c'est le cas elle va intervertir les éléments que contiennent ces deux cases et ainsi simuler le déplacement de nos pièces.

2.2 Utilisation de l'écouteur

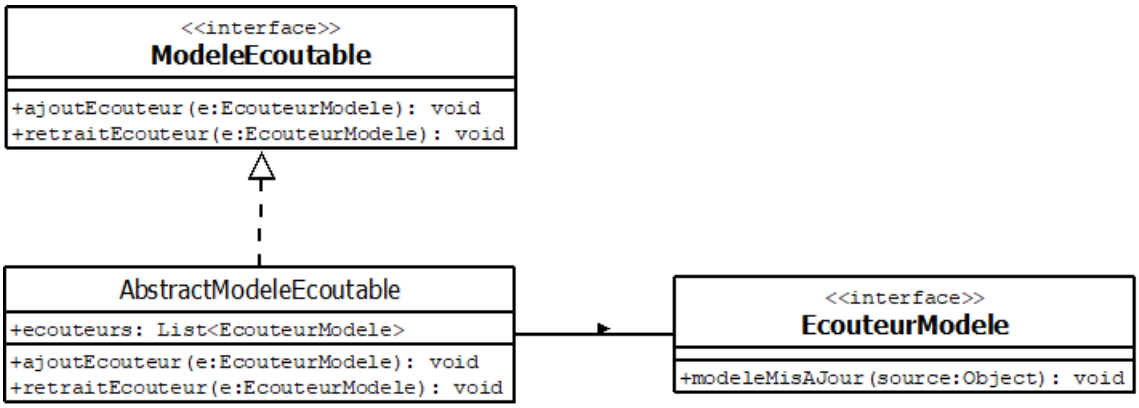

Figure 3: *D*iagramme du package util

Afin que les parties de notre code soient indépendantes mais que notre modèle soit prévenu des changements d'états de notre jeu, ce dernier doit posséder un écouteur Les éléments nécessaires à la mise en place de cet écouteur sont situés dans le fichier util du package models. Le modèle de l'écouteur utilisé et le même que celui vu en TP.

2.3 Fonctionnement de l'interface graphique

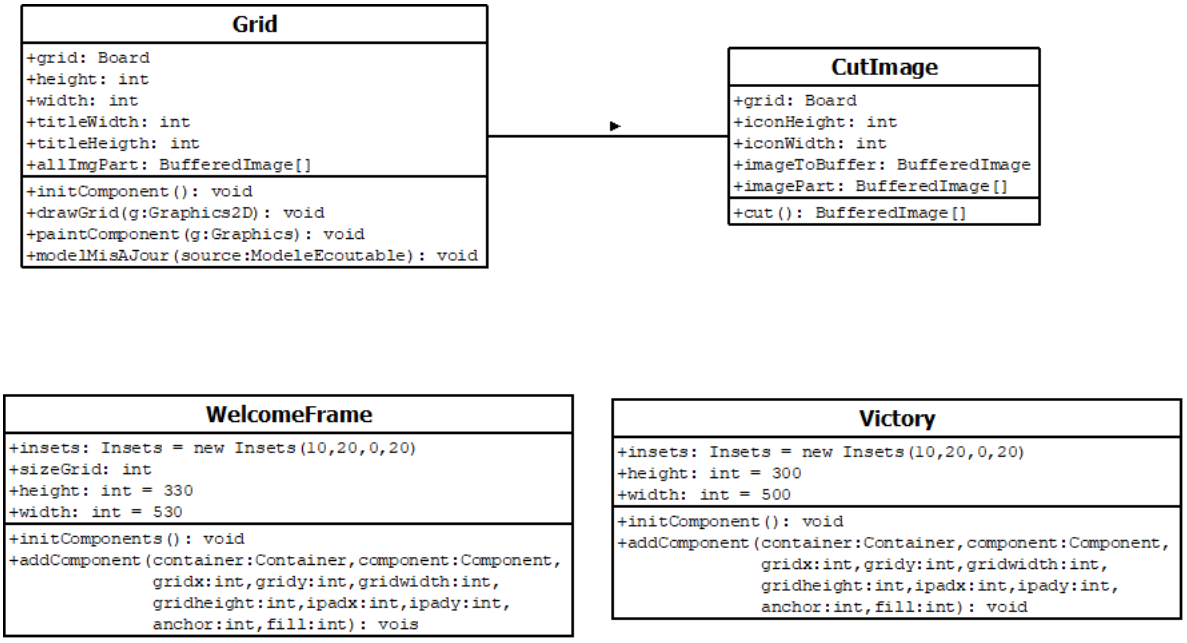

Figure 4: *D*iagramme du package views

Pour réaliser l'interface graphique de notre jeu, nous avons utilisé la bibliothèque SWING. La représentation graphique de notre plateau de jeu se fait dans la classe Grid qui est définie par un attribut de type Board qui est notre instance du plateau de jeu, par height et width, la hauteur et la largeur de notre puzzle qui vont dépendre de l'image sur laquelle on va jouer. Ces derniers permettent de déterminer également les largeurs et les hauteurs de chaque case du puzzle qui nous serviront par la suite. Le constructeur de cette classe met un écouteur sur notre grille de jeu et initialise ces attributs et le contenu des cases en associant chacune au morceau de l'image qui lui correspond.

Pour pouvoir couper une image donnée en plusieurs images et être capable des les associer à nos différentes cases du plateau, nous utilisons la classe CutImage. Cette classe posséde une méthode cut qui parcourt l'image donnée et la coupe en plusieurs images en fonction de la taille de notre grille. Les parties d'images ainsi formées sont stockés dans une liste appelée imagePart

et pourront aisni être utilisées pour initialiser notre grille. Pour que le jeu soit jouable à la souris on utilise la fonction initComponent qui initialise la dimension de la fenêtre de jeu et qui implémente un écouteur pour la souris capable de récupérer la position du clic d'un joueur.

La méthode paintComponent qui appelle la méthode drawGrid permet de parcourir notre plateau et de créer le rendu graphique de celui-ci. La classe WelcomeFrame est la classe nous permettant d'implémenter la fenêtre de lancement du jeu. On pourra choisir dans cette fenêtre la taille du puzzle souhaitée, l'image avec laquelle nous voulons jouer et comportera un bouton pour démarrer le jeu. La classe Victory est la classe permettant la création d'une fenêtre qui s'ouvre lorsque le joueur a gagné. Cette fenêtre proposera notamment deux boutons au joueur, l'un pour rejouer et l'autre pour quitter la partie.

2.4 Mise en place du contrôleur

Le contrôleur de notre jeu sera le clavier, et est réalisé dans la classe KeyBoard. Cette classe a pour attribut notre plateau de jeu de type Board et un booléen appelé keyActive qui va nous permettre d'empêcher la répétition d'un appel à une méthode. Elle possède une méthode keyPressed qui permet déplacer la case vide dans une direction en fonction de la touche utilisée par le joueur.

Ainsi notre jeu peut se jouer au clavier, auquel cas c'est la case vide qui est déplacée, mais aussi à la souris où dans ce cas il faudra cliquer sur une pièce adjacente à la case vide pour la déplacer vers celle-ci.