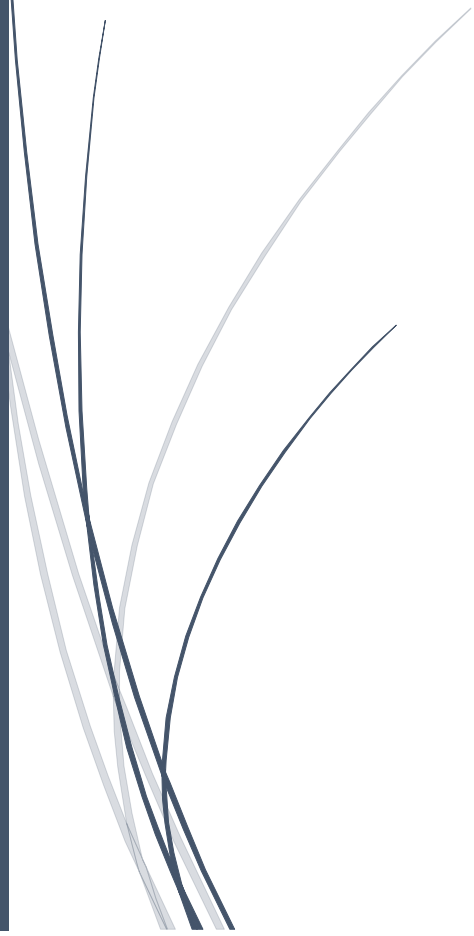
30/05/2016

Livrable du jeu des 6 couleurs

Mission :

Créer un jeu des 6 couleurs fonctionnel sous Java.



Binôme : Pierre-Olivier Carli
Louis Jourdan

Table des matières

Introduction

Le jeu des 6 couleurs est un jeu au tour par tour sur plateau avec des cases colorées de 6 couleurs différentes ou les joueurs débutent chacun dans un coin du plateau et doivent étendre leurs territoires en choisissant des couleurs à assimiler parmi les cases adjacentes. Le gagnant est celui qui a le plus grand territoire. Toute les cases contrôlées par un joueur sont de la dernière couleur choisie par ce joueur et ne peut pas être choisi par d'autres joueurs dans le même tour. La partie prends fin lorsqu'un joueur contrôle plus de la moitié des cases ou quand il n'y a plus de cases non contrôlées.

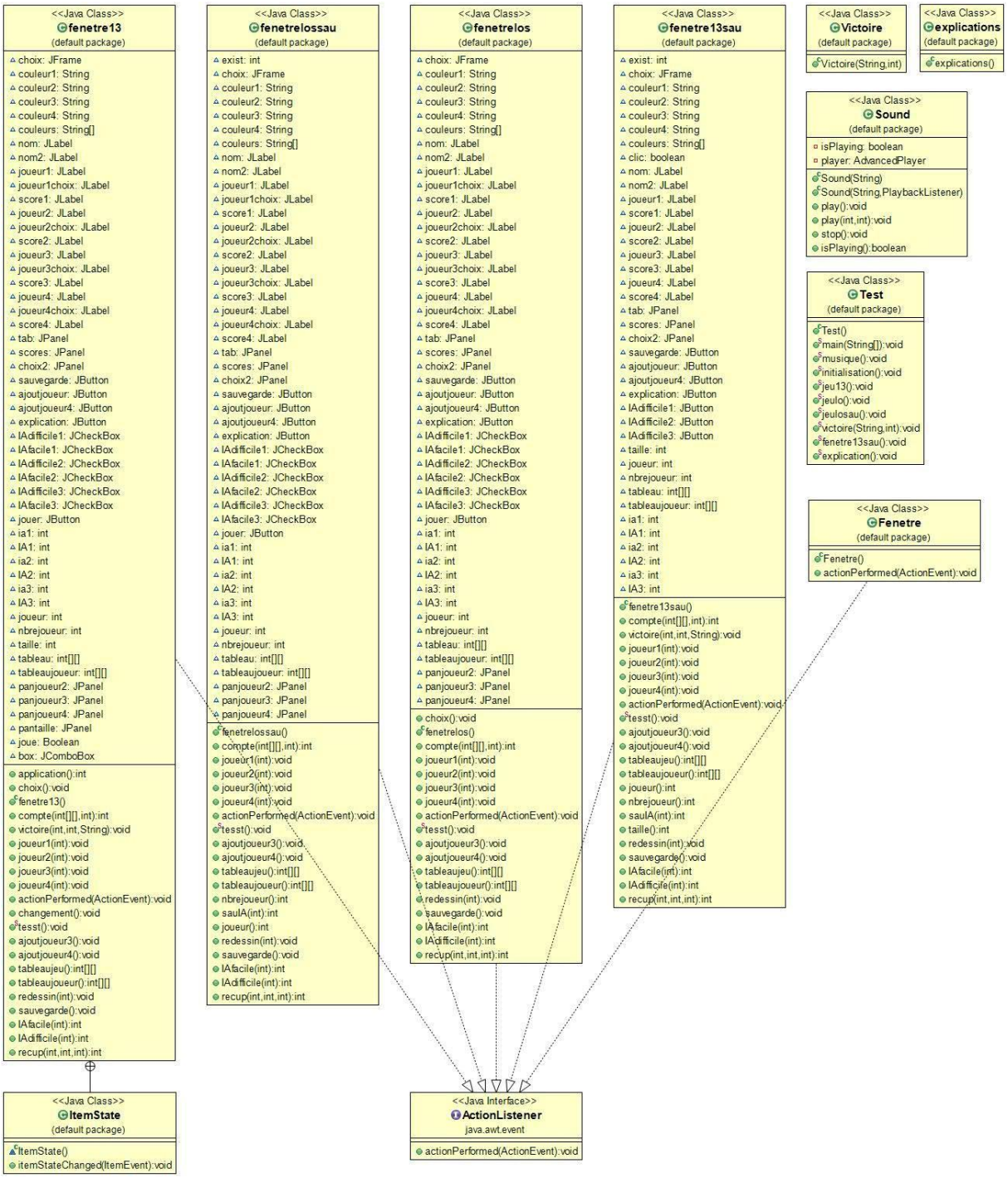
Modélisation

Afin de réaliser ce jeu, nous avons créé 9 classes principales :

- Test : il s'agit de la classe qui va lancer le jeu. En effet, celle-ci est constituée de plusieurs void permettant de lancer d'autres classes. Cette méthode permet ainsi de pouvoir jouer plusieurs parties en même temps.
- Fenetre : cette classe utilise JFrame pour créer une fenêtre visible. Elle utilise ensuite plusieurs JButton, qui permettrons de renvoyer à la classe Test et de lancer d'autres classes.
- Fenetre13 : cette classe est celle du jeu carré. Elle utilise JFrame et ActionListener, qui permettra d'utiliser les JButton. Lors de l'ouverture de la fenêtre, une autre fenêtre est créée, celle-ci permet de modifier les options du jeu. Les tableaux permettant de faire évoluer les territoires des joueurs et les couleurs affichées sont créés par des fonctions internes à la classe.
    - La classe ItemState est instanciée dans cette classe : celle-ci permet de récupérer l'état du menu déroulant permettant de choisir la taille de la grille.
- Fenetre13sau : cette classe est exactement la même que la précédente, cependant, la création des tableaux de jeu et des joueurs se fait depuis un fichier de sauvegarde.
- Fenetrelos : cette classe fonctionne globalement comme celle du jeu carré, cependant, le tableau créé diffère dans sa forme. La méthode permettant de le créer est donc différente.
- Fenetrelossau : tout comme fenetre13sau, celle-ci récupère ses tableaux dans un fichier de sauvegarde.
- Victoire : cette classe crée une fenêtre affichant le gagnant et le nombre de cases lorsqu'un joueur gagne la partie.
- Explications : cette classe lance une fenêtre toute simple sans fonctionnalités, permettant d'avoir des explications sur le jeu.
- Sound : cette classe a été récupérée sur Internet, elle permet de lire une musique et d'utiliser l'API javazoom.

Voici l'UML de notre code source : celui-ci permet de repérer les variables de chaque classe ainsi que les classes utilisées.

On remarque que les classes sont indépendantes, cela permet de faire plusieurs jeux à la fois.



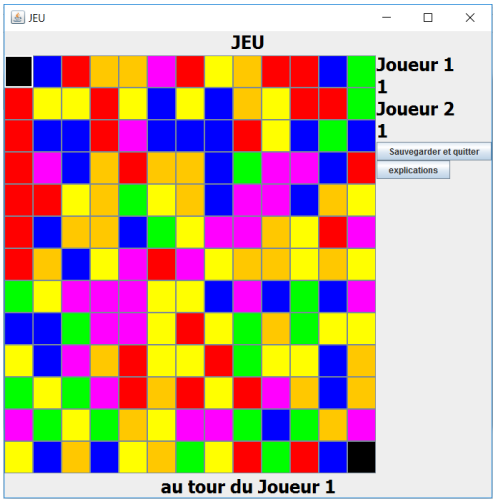
Choix et description des algorithmes utilisés

Chaque type de jeu fonctionne de la même façon :

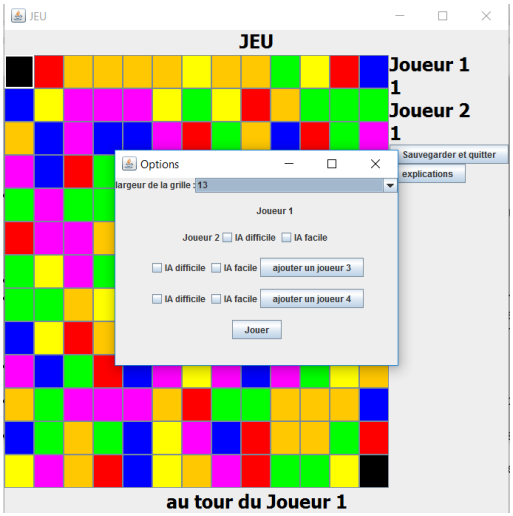
- Création de deux tableaux, l'un aléatoire rempli avec des chiffres de 1 à 6, permettant l'attribution des couleurs. Puis création d'un tableau rempli de 0 de la même taille. On remplace ensuite les 0 en haut à gauche et en bas à droite par 1 et 2, permettant de donner aux joueurs leur case initiale.
- La fenêtre d'options permet de modifier ces informations, en ajoutant des joueurs ou en les transformant en intelligences artificielles.
- Une fois ces tableaux créés, le jeu fonctionne avec quatre fonctions, une par joueur.
    - Chaque joueur doit sélectionner une couleur, lorsque la couleur est sélectionnée (clic sur un bouton), on lance sa fonction, qui va modifier les cases lui appartenant, et lui donner les cases collées qui ne sont pas encore à lui, mais qui sont de la bonne couleur.
    - Si le joueur suivant est une Intelligence Artificielle, alors on appelle la fonction correspondante (facile ou difficile) qui va choisir la couleur à la place du clic, et lancer la fonction du joueur suivant.
    - A chaque fois qu'un joueur joue, on compte le nombre de cases portant son numéro dans le tableau correspondant. Si celui-ci est supérieur au nombre de cases divisées par le nombre de joueurs, alors il a gagné et une fenêtre s'ouvrira.
- L'intelligence artificielle facile fonctionne simplement : on prend un nombre aléatoire, qui lui attribuera une couleur. On réalise cette opération jusqu'à trouver un nombre aléatoire qui n'est pas représentatif de la couleur d'un autre joueur.
- L'intelligence artificielle difficile fonctionne différemment. Elle crée un tableau de six 0. Puis elle parcourt la grille pour savoir combien de cases va lui faire gagner chaque couleur. La case correspondante dans le tableau est alors modifiée, et on prend la plus grande, si celle-ci n'appartient pas à un autre joueur.
- A tout moment, l'utilisateur peut sauvegarder sa partie :
    - Le tableau correspondant aux couleurs affichées va être sauvegardé dans un premier fichier texte, une case par ligne.
    - Le tableau correspondant aux territoires des joueurs est sauvegardé de la même façon, dans un autre fichier.
    - Enfin, dans un fichier autre, on enregistre la taille de la grille, le nom des joueurs, le type d'intelligence artificielle utilisée si besoin.
    - Enfin, le chargement de la sauvegarde se fait en créant deux tableaux vides. Ceux-ci se remplissent en lisant le tableau.

Fonctionnalités et ajouts

- Version graphique complète réalisée avec Swing : pas d'affichage console prévu. Swing permettait d'utiliser plus facilement des boutons à la place des cases dans le jeu.



- Choix du nombre de joueurs, de 2 à 4 : cela se fait en remplaçant le nom (vide initialement) des joueurs 3 et 4. S'ils ont un nom, alors ils sont considérés comme joueurs, sinon, on ne rentre jamais dans leur fonction.



- Deux formes de grilles disponibles : carrée et losange.

- Taille de la grille carrée modifiable : de 5x5 à 50x50 : cela se fait grâce à un menu déroulant, dont on récupère la valeur grâce à une classe annexe (ItemState). Une fois l'état récupéré, on le traduit en String puis en Integer pour l'utiliser dans la création des tableaux et des scores nécessaires à la victoire.
- Plusieurs niveaux d'IA : IA facile qui choisit une couleur aléatoire et IA plus difficile qui choisit la couleur lui permettant de capturer le plus de cases.
- Possibilité de sauvegarder la partie dans un fichier externe et de reprendre la partie plus tard.
- Ajout d'une musique d'ambiance (Nova, de Ahrix) dans le jeu : cette lecture de musique se fait dans le main. Il n'est pas utile dans notre programme d'utiliser un thread dédié, car le jeu ne dépend pas du main. On a utilisé un lecteur de musique développé par d'autres personnes, dont l'API est trouvable en ligne assez facilement.

Conclusion

Finalement, nous avons fait le programme en utilisant Swing. Cela était selon nous un bon choix car il a permis plus de simplicité dans le jeu (boutons faisant office de cases). Nous avons développé le jeu en apprenant à nous servir des outils, alors que nous aurions pu attendre un peu plus afin de pouvoir le développer en MVC. Cependant, ce programme nous a permis de développer nos compétences en java, et en travail en équipe.