RAPPORT DE PROJET
2048

UE LIFAP7 : Algorithmique Et Programmation Orientée Objet
L3 informatique
Semestre de printemps 2022

BERNOT Camille, *p1908800*
FIETIER Loris, *p1805561*

LADJAL Hamid
Salima HASSAS ADJALI

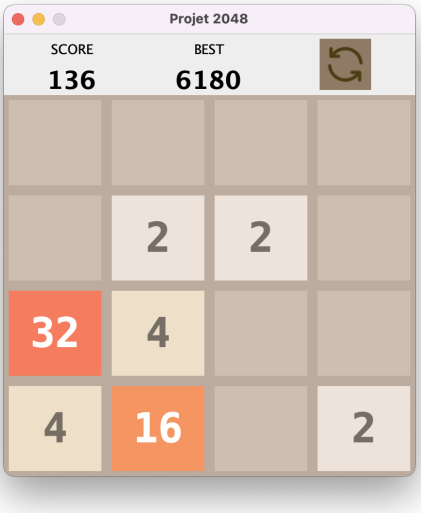
Listes des fonctionnalités et contributions

Fonctionnalité	Contributeur principal
Améliorations du rendu graphique (coloration des cas	BERNOT Camille
Mouvement et fusion des cases	BERNOT Camille
Sauvegarde du meilleur score	BERNOT Camille
Menu version de séléction partie solo/duo	FIETIER Loris
Version deux joueurs	FIETIER Loris

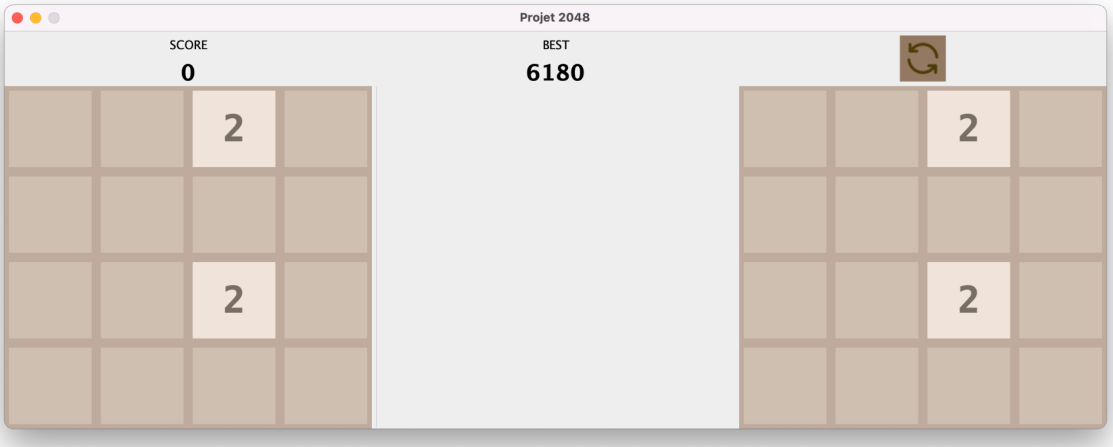
Nous tenons cependant à souligner que dans quasiment toutes les étapes du projet, lorsqu'un membre du binôme intégrait son code au repo distant, l'autre membre du binôme corrigeait les éventuelles imperfections et participait activement à l'amélioration du projet.

Copies d'écran

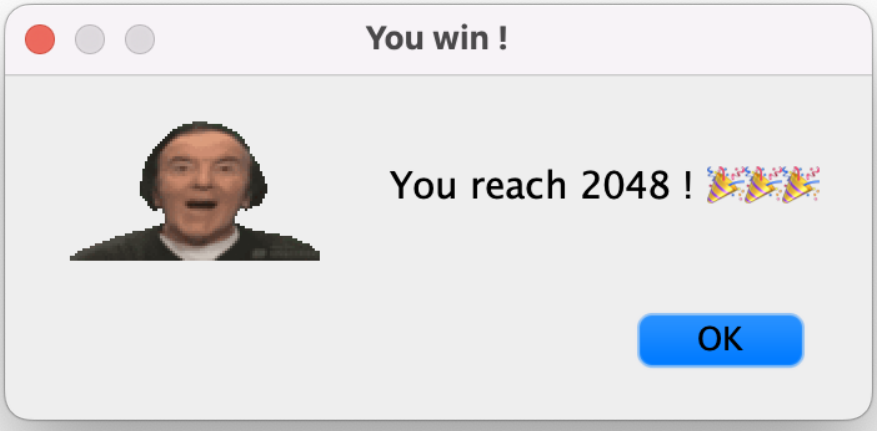
Menu au démarrage du jeu.



Partie (1 joueur) en cours.



Partie en mode deux joueurs en cours.



Pop-up affichée lors de la victoire d'un joueur (lorsqu'une case atteint la valeur 2048).

Quelques détails techniques

Nous avons décidé de procéder d'une manière un petit peu particulière pour effectuer le déplacement des cases. Comme précisé dans l'énoncé, nous avons bien une fonction `void action(Direction direction)` dans la classe `Jeu` et une fonction `boolean deplacer()` dans la classe `Case`.

`boolean deplacer()`

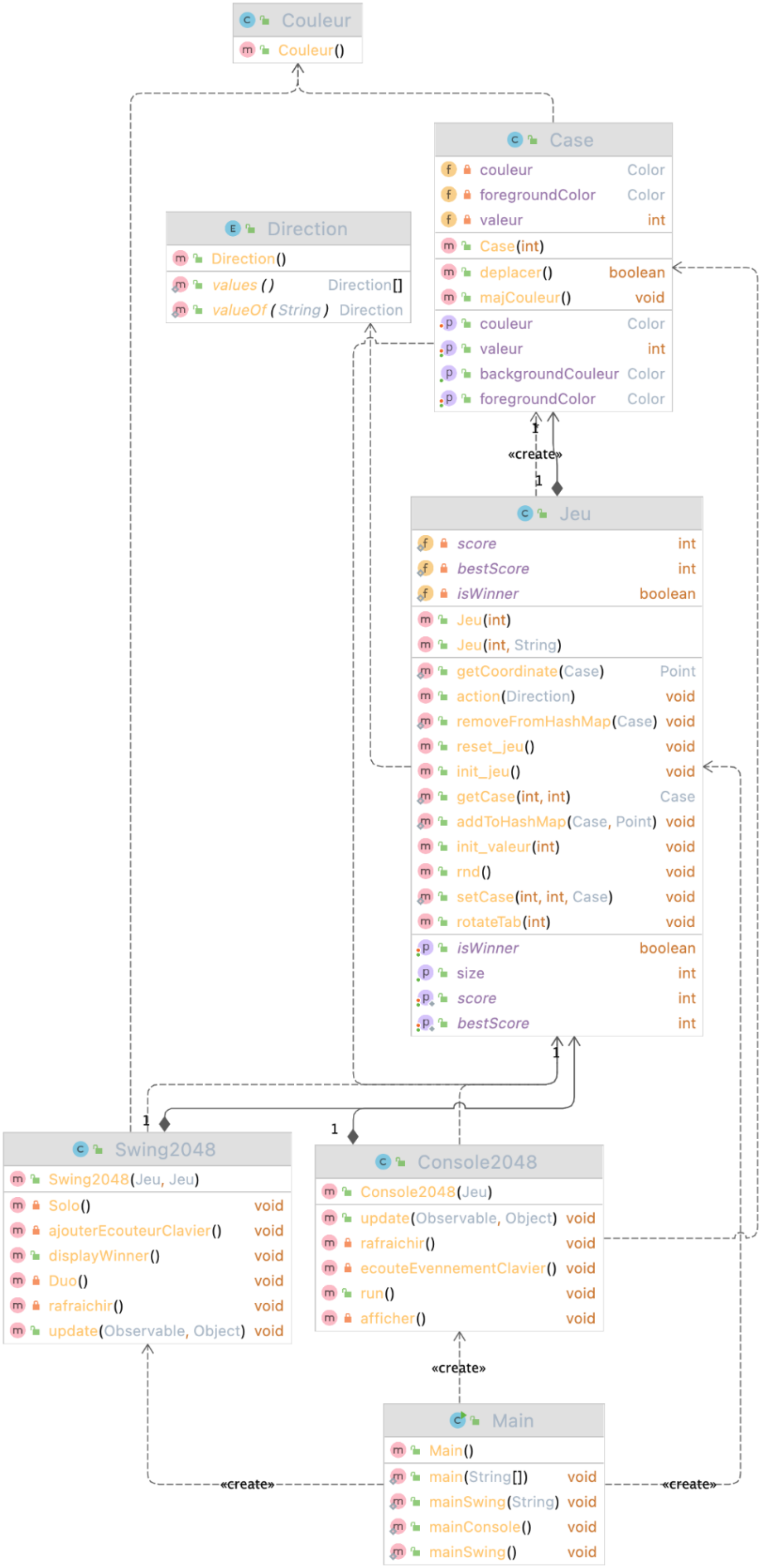
Cette fonction permet de déplacer une seule case. Elle ne prend pas de direction en paramètre, car nous avons implémenté celle-ci de façon à ce que la case se déplace **toujours vers le haut** (nous utilisons donc un petit subterfuge dans `action()` de la classe `Jeu` pour palier à cela). Nous récupérons donc les coordonnées de la case dans la hashmap de la classe `Jeu` puis nous regardons uniquement son voisin haut. Tant que ce dernier est nul, alors nous prenons sa place. Si la valeur ce dernier est égale à la valeur de la case traitée, alors nous créons une nouvelle case dont la valeur est la somme de la case traitée et de son voisin.

`void action(Direction direction)`

Dans cette fonction de la classe `Jeu`, nous commençons tout d'abord par réaliser une ou plusieurs rotations du tableau de cases de la classe `Jeu` (selon la direction) afin qu'un mouvement vers la direction souhaitée corresponde à un mouvement vers le haut. Nous parcourons ensuite le tableau de cases et appelons `deplacer()` sur toutes les cases non nulles. Nous terminons cette fonction en réalisant les rotations inverses du tableau de cases afin de le remettre dans sa configuration initiale.

Diagramme UML

Parmi les diagrammes proposés, nous hésitions entre un diagramme de cas d'utilisation et un diagramme de classes et avons finalement décidé d'intégrer au projet un diagramme de classes. En effet, à notre sens, montrer le système du point de vue des acteurs avait moins d'intérêt que de montrer la structure interne étant donné que les cas d'utilisation sont accessibles et peuvent être découverts lors de l'utilisation du programme.
Le diagramme de classes est donc plus adéquat et offre une vision globale des liens et comportements entre les différentes classes. Cela permet également de repérer rapidement les attributs et méthodes de chaque objets.


Couleur
Couleur()
Case
couleur Color
foregroundColor Color
valeur int
Case(int)
deplacer() boolean
majCouleur() void
couleur Color
valeur int
backgroundCouleur Color
foregroundColor Color
Direction
Direction()
values() Direction[]
valueOf(String) Direction
1
«create»
1
Jeu
score int
bestScore int
isWinner boolean
Jeu(int)
Jeu(int, String)
getCoordinate(Case) Point
action(Direction) void
removeFromHashMap(Case) void
reset_jeu() void
init_jeu() void
getCase(int, int) Case
addToHashMap(Case, Point) void
init_valeur(int) void
rnd() void
setCase(int, int, Case) void
rotateTab(int) void
isWinner boolean
size int
score int
bestScore int
1
1
1
Swing2048
Swing2048(Jeu, Jeu)
Solo() void
ajouterEcouteurClavier() void
displayWinner() void
Duo() void
rafraichir() void
update(Observable, Object) void
Console2048
Console2048(Jeu)
update(Observable, Object) void
rafraichir() void
ecouteEvennementClavier() void
run() void
afficher() void
«create»
Main
Main()
main(String[]) void
mainSwing(String) void
mainConsole() void
mainSwing() void
«create»
«create»