Thomas BROGGI

Maud CHOLEZ

Sophie DA COSTA

RAPPORT DE PROJET 2048

*UFD61 Mathématiques et informatique*

*Ingénierie Logicielle*

*Enseignant : M. CASTAGNOS Sylvain*

*L3 MIASHS – Parcours Sciences Cognitives*

*2016 – 2017*

*Sophie DA COSTA  
Maud CHOLEZ  
Thomas BROGGI*

Sommaire

[INTRODUCTION 3](#_Toc480906644)

[RECETTE INFORMATIQUE 4](#_Toc480906645)

[Achievements à atteindre 4](#_Toc480906646)

[Achievements atteints ou en cours 4](#_Toc480906647)

[Jeu en solo dans la console Netbeans 4](#_Toc480906648)

[Jeu en solo avec interface graphique au clavier 4](#_Toc480906649)

[Affichage fluide dans l’interface graphique avec des Threads 5](#_Toc480906650)

[Possibilité de demander de l’aide à une I.A. pour le prochain coup 5](#_Toc480906651)

[Possibilité de laisser l’I.A. jouer toute seule jusqu’à 2048 5](#_Toc480906652)

[Possibilité de recharger la partie en cours 6](#_Toc480906653)

[Possibilité de voir le classement des joueurs de l’application 6](#_Toc480906654)

[Jeu multi-joueurs 7](#_Toc480906655)

[Planning de notre projet 7](#_Toc480906656)

[DESCRIPTION DE L’APPLICATION 8](#_Toc480906657)

[Installation de notre application 8](#_Toc480906658)

[Retour sur l’interface et ses fonctionnalités 9](#_Toc480906659)

[DIAGRAMMES 12](#_Toc480906660)

[Diagramme de cas d’utilisation 12](#_Toc480906661)

[Diagramme de classes 12](#_Toc480906662)

[Diagramme de séquence 12](#_Toc480906663)

[REPOSITORY 14](#_Toc480906664)

[DOCUMENTATION 16](#_Toc480906665)

[RETROSPECTIVE ET CONCLUSION 18](#_Toc480906666)

# INTRODUCTION

Dans le cadre de notre 3ème année de licence en Mathématiques et Informatique Appliqués aux Sciences Humaines et Sociales (MIASHS), parcours Sciences Cognitives, nous avons eu pour projet de créer le jeu 2048. Ce projet devait être réalisé dans le cadre de l’UFD41 Mathématiques et Informatique, avec comme enseignant M. CASTAGNOS Sylvain, et il représente l’intégralité de l’examen de cette matière.

Tout d’abord, nous allons nous intéresser à ce qu’est le jeu 2048. Il a été conçu en mars 2014, il est disponible sur ordinateur, tablette et smartphone. Il s’agit d’un jeu représenté par une grille de 16 cases. Le but du jeu est de réunir des tuiles de même valeur, afin d’atteindre la tuile portant la valeur de 2048. Les chiffres disponibles sont des puissances de 2 : 2, 4, 8, 16, 32, …, 2048, … Après chaque déplacement réalisé, une nouvelle tuile est générée, avec comme chiffre 2 ou 4. Une fois l’objectif atteint, il est possible de stopper la partie ou alors de continuer.

Le projet s’est déroulé en deux principales étapes :

* La mise en place du projet : Création de différents diagrammes pour gérer les différentes classes et élaboration du planning pour gérer les étapes de la conception.
* La conception des différentes classes, méthodes, fonctions nécessaires pour atteindre les achievements.

Dans ce rapport, nous allons décrire les étapes qui nous ont permis de mener à bien ce projet. Ce qui permettra de comprendre nos choix et de faciliter leur expérience sur notre application.

# RECETTE INFORMATIQUE

## Achievements à atteindre

Dans un premier temps, nous allons revoir les achievements que nous avions à réaliser, dans l’ordre de notre choix.

1. *Jeu en solo dans la console Netbeans, qui a déjà été réalisé en TP noté de L2,*
2. Jeu en solo avec interface graphique au clavier,
3. Affichage fluide dans l’interface graphique avec des Threads,
4. Possibilité de demander de l’aide à une I.A. pour le prochain coup,
5. Possibilité de laisser l’I.A. jouer toute seule jusqu’à 2048 (ou interruption à l’aide d’un bouton stop),
6. Possibilité de recharger la partie qui était en cours lors du précédent lancement du jeu,
7. Possibilité de voir votre classement parmi les joueurs de l’application,
8. Jeu multi-joueurs.

## Achievements atteints ou en cours

### Jeu en solo dans la console Netbeans

En ce qui concerne le premier achievement de ce projet, nous avons décidé de garder l’intégralité du code fourni dans la correction du TP noté de l’année passée. En effet, nous avons analysé l’ensemble des classes données, et nous avons bien compris les méthodes utilisées, alors nous n’avions aucune raison de ne pas l’utiliser. Une fois que nous avions pris connaissance de la correction, nous avons procéder à l’écrire de l’API qui la concernait. Cela nous a permis d’être au clair avec l’ensemble du contenu.

### Jeu en solo avec interface graphique au clavier

Cet achievement est le premier que nous avons décidé de faire. Il nous paraissait important de le terminer. Pour jouer à 2048, c’est plus simple d’avoir une interface graphique nous permettant de savoir où l’on se trouve dans la partie plutôt que de s’appuyer sur ce qui est affiché dans la console. Cet achivement a été réalisé par Thomas. Nous avons décidé que le choix des déplacements serait effectué au clavier, à l’aide des flèches directionnelles gauche et droite. La mise en place de l’interface graphique s’est faite à l’aide du logiciel JavaFX Scene Builder. Elle est composée d’une grille, sur laquelle nous venons placer les images correspondantes aux 16 cases présentes dans l’ensemble symbolisant la grille. Une fois que la base de l’interface était faite, il nous a fallu faire le lien entre les classes et les méthodes fournies dans l’étape précédente. Nous avons alors mis en place le déroulement d’une partie, en actualisant l’affichage en fonction des cases qui devraient être présentées. C'est-à-dire qu’après chaque déplacement nous appelons une méthode *ajoutImageCase(Grille g),* qui permet d’afficher les images là où elles doivent être dans la grille, en fonction de leurs coordonnées.

### Affichage fluide dans l’interface graphique avec des Threads

La seconde étape que nous voulions accomplir était celle de l’intégration des Threads dans l’interface graphique. Ainsi, ce sont Sophie et Thomas qui se sont penchés dessus. Mais après plusieurs essais infructueux, nous avons décidé de la laisser de côté, pour ne pas perdre trop de temps et afin de pouvoir avancer sur d’autres points. Au terme de ce projet, il semblerait que le fait que nous n’avons pas réussi cette étape serait lié à la méthode que nous avons utilisée pour mettre en place l’interface graphique du niveau précédent. Nous avons réalisé l’affichage des cases en fonction de coordonnées, et non pas en fonction de pixels. Ce qui a induit des difficultés lorsque nous avons voulu mettre en place les Threads.

### Possibilité de demander de l’aide à une I.A. pour le prochain coup

Pour cette partie, nous avons commencé par nous demandé quelle(s) heuristique(s) nous devions utiliser. Nous sommes arrivés aux deux possibilités suivantes :

1. L’IA effectue un coup qui permet d’obtenir le moins de cases possibles dans la grille,
2. L’IA effectue un coup qui permet de fusionner les cases ayant les valeurs les plus grandes.

Après concertation, nous avons conclue que pour la seconde heuristique proposée, l’utilisateur n’avait pas besoin de l’aide d’une intelligence artificielle pour effectuer le déplacement permettant de fusionner les cases ayant les valeurs les plus élevées. En effet, si l’utilisateur demande de l’aide, c’est, normalement, parce qu’il ne sait pas quel coup effectuer pour se sortir d’une situation difficile. Ainsi, le fait de conseiller le déplacement qui générera le moins de cases par la suite semblait être plus approprié. Dans le but d’améliorer davantage notre heuristique, nous l’avons programmée sur une profondeur de deux et nous avons privilégié les déplacements vers le haut et vers la gauche. Le déroulement du programme est donc le suivant : nous établissons une copie de la grille de la partie en cours afin de pouvoir effectuer les déplacements de tests et analyser les résultats associés. Ensuite, l’IA fait les tests sur quatre couples différents de déplacements : (gauche, gauche), (gauche, haut), (haut, gauche) et (haut,haut), ces tests se trouvent dans la méthode testAide(), pour chaque test on récupère le nombre de cases qui en ressort, et on compare lequel est le plus faible afin de ne garder que le déplacement le plus intéressant pour notre heuristique. Enfin, on effectue le déplacement approprié.

### Possibilité de laisser l’I.A. jouer toute seule jusqu’à 2048

Une fois que l’aide de l’intelligence artificielle pour le prochain coup est mise en place, le déroulement d’une partie entière n’est pas compliqué à mettre en place car il va utiliser ce qui a été fait précédemment.

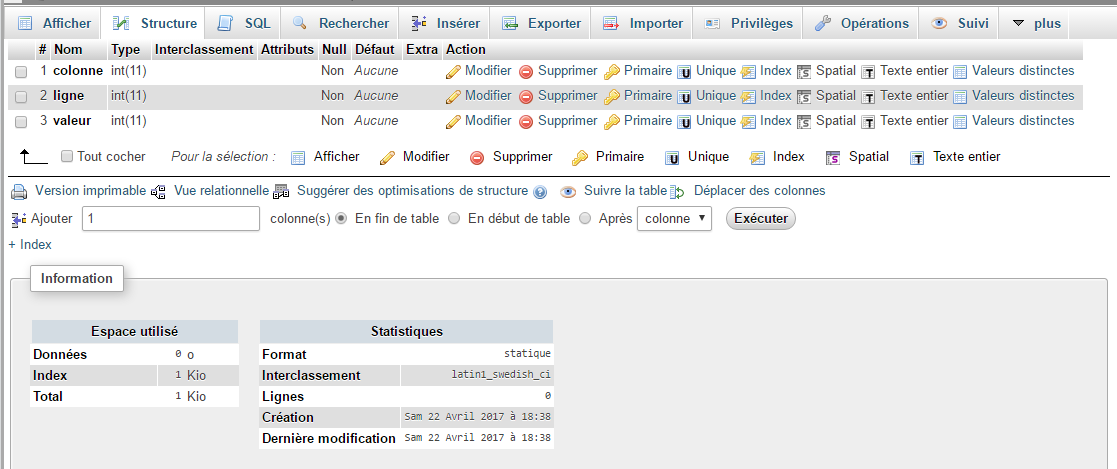
Ainsi, cet achievement est représenté par la méthode deplacementCompletIA() de la classe InterfaceController.java, le déroulement est simple le déplacement à réaliser est déterminer grâce à la méthode testIA() – présente dans la même classe – puis une nouvelle case est générée. Dans le cas où la case n’est pas générée, la méthode gameOver() est appelée, dans le cas contraire c’est la méthode victory() qu’il faut appelée. Ensuite, cette méthode est appelée dans la méthode jeuxIA() qui permet le déroulement complet de la partie lorsque le joueur le demande en appuyant sur le bouton associé (IA dans l’interface).

Nous avons un problème pour cet achievement car une fois que l’on appuie sur le bouton pour démarrer la partie jouée par l’intelligence artificielle, le message de défaite ou de victoire s’affiche directement. La mise à jour de la grille après chaque coup ne se fait pas. Au terme de notre projet, nous n’avons pas réussi à trouver la raison de ce problème. De plus, nous n’avons pas prévu le bouton pour l’interruption.

### Possibilité de recharger la partie en cours

Cet achievement est atteint grâce à Sophie et Thomas. Après avoir rencontré quelques problèmes dans la mise en place de la sérialisation, nous avions laissé le code de côté. Pour finalement y revenir est réussir à la boucler plus facilement. Mais lorsque Thomas a repris cet achievement, il a décidé de ne plus utiliser la méthode de sérialisation mais plutôt de se baser sur la sauvegarde dans une base de données. Cela permet d’enregistrer les valeurs contenues dans la grille de la partie en cours et de les réutiliser lorsque le joueur reprendra sa partie.

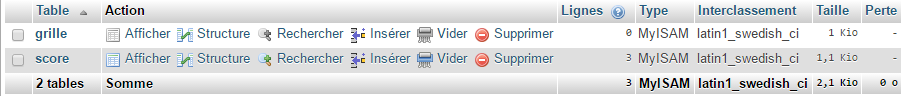
Voici le détail de ce contient la table pour la sauvegarde d’une partie :



### Possibilité de voir le classement des joueurs de l’application

Pour atteindre cet objectif, nous avons à nouveau utilisé la base de données qui avait été créée précédemment. Nous avons donc mis en place une nouvelle table permettant de recevoir les informations d’un joueur à la fin de sa partie.

Voici le détail du contenu de cette table :



Quand le joueur a terminé de jouer, qu’il ait gagné ou qu’il ait perdu, il peut enregistrer son score afin qu’il apparaisse dans le classement des joueurs de l’application. Pour cela, le programme lui demande de préciser son pseudo et ensuite il récupère le score de la partie pour le stocker dans les attributs associés.

### Jeu multi-joueurs

Nous avons décidé dès le début du projet que nous ne ferions pas cet objectif en priorité, car nous savions que nos compétences en programmation Java n’étaient pas suffisamment grandes pour y parvenir.

## Planning de notre projet

Au commencement de notre projet, nous avons établi un planning regroupant les achievements à réaliser et les données qui devaient être présentées dans notre rapport. Nous avons tenté de définir des dates limites pour la réalisation de chacune de ces tâches. Nous n’étions pas tout à fait sûrs de l’ordre dans lequel nous allions les traiter, donc ce n’était pas forcément facile de déterminer des dates, mais nous l’avons tout de même fait pour pouvoir avoir un regard sur l’avancement de notre projet.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Planning projet 2048** | | | | | | | |
| N° | Tâche : | Durée (en h) : | Commencée le : | *Devrait être terminée le :* | Terminée le : | Fait par : | Respect du délai : |
| 1 | Diagramme de cas d'utilisation | 3 | 09.01.2017 | *06.02.2017* | 13.02.2017 | Sophie |  |
| 2 | Diagramme de classes | 2 | 09.01.2017 | *16.01.2017* | 16.01.2017 | Sophie |  |
| 3 | Diagramme d'interaction | 3 | 09.01.2017 | *06.02.2017* | 13.02.2017 | Sophie |  |
| 4 | Création d'un répertoire via GitHub | 0,5 | 09.01.2017 | *09.01.2017* | 09.01.2017 | Sophie, Maud |  |
| 5 | Création du Trello | 0,5 | 09.01.2017 | *09.01.2017* | 09.01.2017 | Sophie, Maud |  |
| 6 | Retour sur le jeu en solo via la console : décision de le garder ou non | 2 | 15.01.2017 | *30.01.2017* | 23.01.2017 | Tout le monde |  |
| 7 | Jeu en solo avec interface graphique au clavier | 6 | 30.01.2017 | *13.02.2017* | 26.03.2017 | Thomas |  |
| 8 | Affichage fluide de l'interface à l'aide de Threads | 6 | 13.02.2017 | *27.02.2017* | abandonné | Sophie, Thomas |  |
| 9 | Aide d'une I.A. pour le prochain coup | 10 | 21.04.2017 | *17.04.2017* | 23.04.2017 | Thomas |  |
| 10 | Partie jouée intégralement par l'I.A., avec possibilité d'interruption | 6 | 23.04.2017 | *24.02.2017* | 23.04.2017 | Thomas |  |
| 11 | Sérialisation | 4 | 06.02.2017 | *06.03.2017* | 20.04.2017 | Sophie, Thomas |  |
| 12 | Possibilité de voir son classement parmi les joueurs | 2 | 21.04.2017 | *25.04.2017* | 21.04.2017 | Thomas |  |
| 13 | Jeu multi-joueurs | 12 | XXX | *24.04.2017* | XXX | XXX |  |
| 14 | Mise en place de la documentation de l'application | 3 | 10.04.2017 | *25.04.2017* | 24.04.2017 | Maud |  |
| 15 | Recette informatique | 2 | 10.02.2017 | *25.04.2017* | 25.04.2018 | Maud |  |
| 16 | Transformation du code source en JAR exécutable | 2 | 25.04.2019 | *25.04.2017* | 25.04.2019 | Thomas |  |
| *17* | *Support de présentation pour la soutenance* | *1* |  | *28.04.2017* |  | *Sophie, Maud* |  |
|  |  |  |  |  |  |  |  |
|  |  |  | Délai respecté | Délai dépassé | Non fait |  |  |

# DESCRIPTION DE L’APPLICATION

## Installation de notre application

Afin de pouvoir utiliser notre application, il faut procéder à l’installation de la base de données sur le logiciel easyPHP, puis à la mise en place celle-ci dans l’IDE que nous utilisions, ici nous décrivons la méthode pour le faire avec NetBeans. Voici les étapes à suivre pour y parvenir :

1. Dans EasyPHP : Importer le fichier 2048.sql dans phpMyAdmin, avec comme nom pour la base de données : 2048.
2. Dans le projet sur NetBeans : Aller dans « Librairies »

> Clic droit sur « Librairies » puis « Add Library »

> Choisir « MySQL JDBC Drive » puis « Add Library »

1. Ouvrir l’onglet window > « Services »

> Clic droit sur « Databases »

> « New connection »

1. Dans la fenêtre « New connection Wizard », dans le champ « Driver »

> « MySQL (Connector/J Driver)

> « Next »

> « Host » : « localhost »

> « Port » : « 3306 »

> « Database » : « mysql »

> « User Name » : « root »

> « JDBC URL » : « jdbc:mysql://localhost:3306/2048 »

1. Test de la connexion :

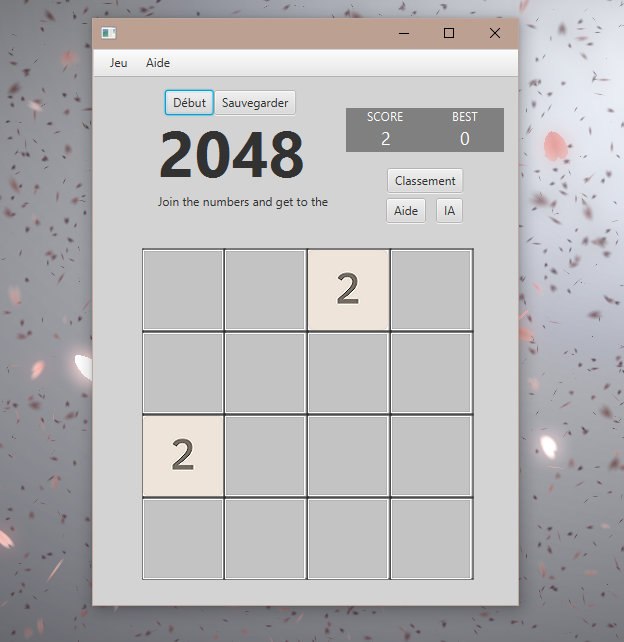
> « Test Connection »

> « Connection succeeded »

Nous n’avons pas réussi à extraire notre application sous l’extension .JAR, lorsque nous avons essayé de le faire, l’affichage des images ne fonctionnait pas, nous n’avons pas eu le temps de régler ce problème car nous l’avons remarqué que tardivement.

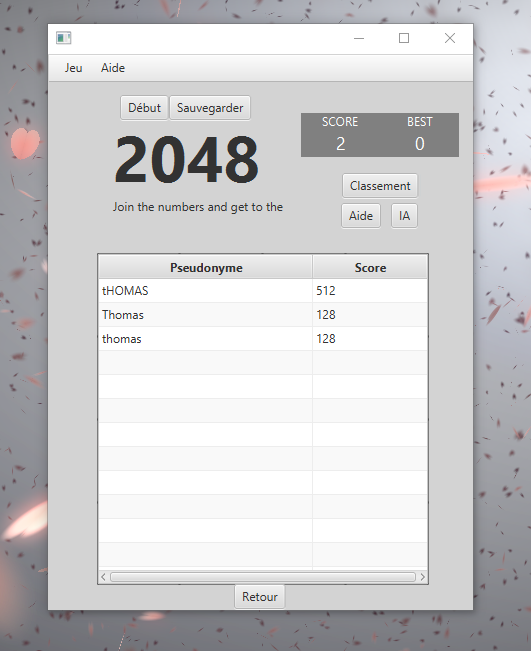
## Retour sur l’interface et ses fonctionnalités

Lorsque le joueur lance l’application, voici ce qui apparait :



Voici une description des fonctionnalités des boutons présents sur cette dernière :

* **Début** : Ce bouton permet de lancer une nouvelle partie.
* **Sauvegarder** : Il s’agit du bouton permettant la sauvegarde de la partie en cours.
* **Classement** : C’est à cet endroit qu’il est possible de visualiser le classement des joueurs de l’application. Voici un aperçu :

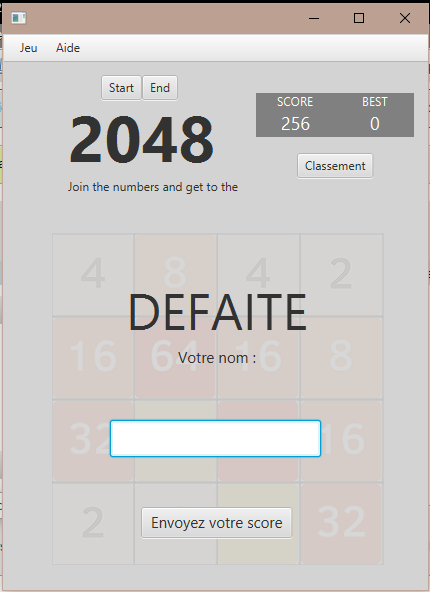


Le bouton Retour permet de revenir au jeu.

* **Aide** : Bouton permettant de demander de l’aide à l’intelligence artificielle pour le prochain coup.
* **IA** : Bouton permettant de lancer une partie complète jouée par l’IA.

L’interface est également dotée d’un emplacement qui affiche le score réalisé pour la partie que nous sommes en train de jouer et à côté l’affichage de notre meilleur score.

A la fin d’une partie, un message de victoire ou d’échec s’affiche et nous propose d’entrer notre nom et d’enregistrer notre score :



# DIAGRAMMES

## Diagramme de cas d’utilisation

Le diagramme de cas d’utilisation représente les fonctionnalités visibles par le joueur quand il utilise notre jeu.

## C:\Users\Maud\Documents\NANCY\2016-2017\L3\S6\Ingénierie logicielle\PROJET\2048-BROGGI-CHOLEZ-DACOSTA-master\2048-BROGGI-CHOLEZ-DACOSTA\rapport\diagramme_utilisation.jpg

## Diagramme de classes

Le diagramme de classes est une représentation statique des éléments qui composent un système. Cela permet de visualiser l’intégralité des classes ainsi que les variables présentes à l’intérieur, les méthodes et les fonctions. Les classes sont reliées entre elles grâce à des associations représentant le rôle que joue une classe vis-à-vis d’une autre.

## C:\Users\Maud\Documents\NANCY\2016-2017\L3\S6\Ingénierie logicielle\PROJET\2048-BROGGI-CHOLEZ-DACOSTA-master\2048-BROGGI-CHOLEZ-DACOSTA\rapport\diagramme_classes.jpg

## Diagramme de séquence

Le diagramme de séquence est une représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique.

*Remarque : Il est possible de retrouver nos diagrammes en format .jpg dans le dossier « Rapport », afin de pouvoir les visualiser plus aisément.*

# REPOSITORY

Afin de gérer au mieux l’organisation de notre projet, nous avons utilisé deux outils conseillés par notre enseignant :

* **GitHub** est un service en ligne permettant d’héberger ses repositories de code. C’est un outil gratuit, que nous avons découvert lors d’un précédent projet. Suite à cette révélation, il était évident pour nous d’utiliser GitHub plutôt que SourceForge.

Voici le lien pour accéder à notre répertoire : <https://github.com/mdchlz/2048-BROGGI-CHOLEZ-DACOSTA>

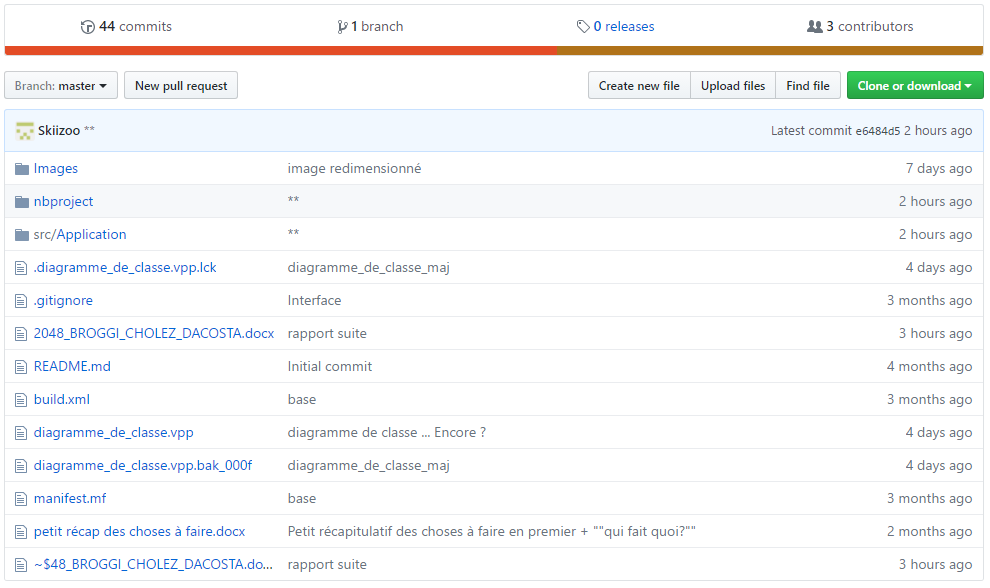


Figure 1 Vue d'ensemble de notre projet sur GitHub

* **Trello** est un outil collaboratif qui nous permet d’organiser nos projets en différentes listes de tâches à effectuer, ou autre. Ainsi, nous retrouvons un tableau de bord dans lequel nous avons créé 5 listes : une regroupant les achievements à faire et précisant dans quel cours nous retrouvions les informations nécessaires. Les autres : pour ce qu’il a à rendre dans le rapport, ce qu’il y a à faire pour la conception, ce qui est en cours et ce qui est terminé. Pour chaque tâche, il est souvent possible de trouver des informations en lien avec celle-ci, et aussi la personne qui doit s’en occuper.

Voici le lien pour accéder à notre tableau de bord : <https://trello.com/b/KeB9cTEo/2048>

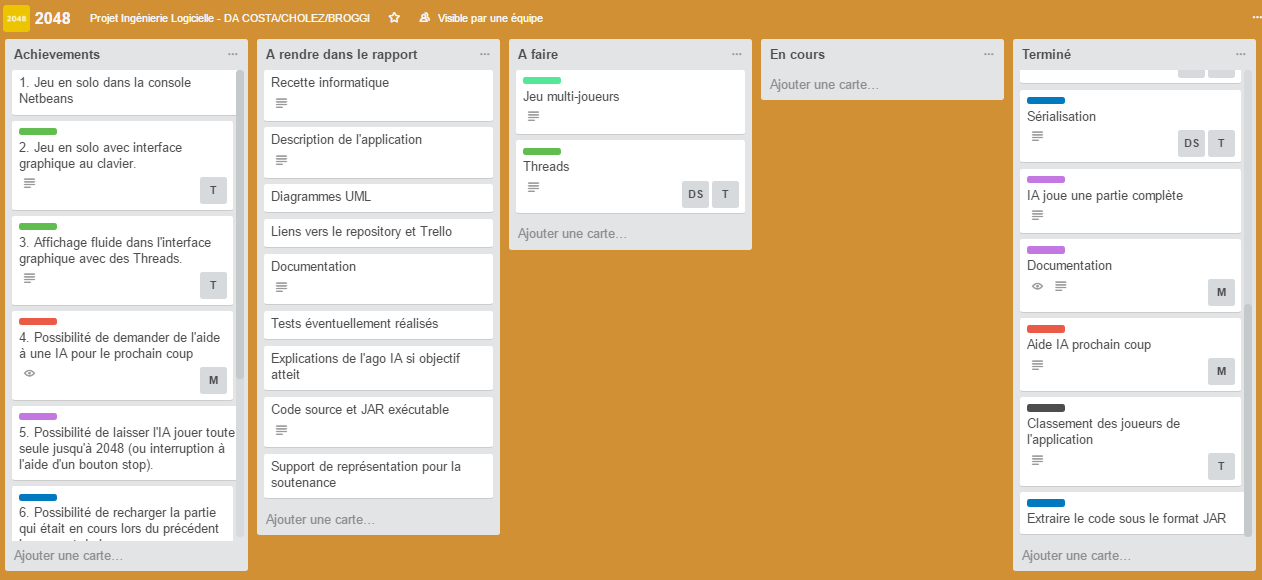


Figure 2 Vue d'ensemble de notre Trello

# DOCUMENTATION

JavaDoc est un outil développé par Sun Microsystems permettant de créer une documentation d’API en format HTML depuis les commentaires présents dans notre code source Java. Il s’agit d’un standard pour la documentation des classes Java et depuis NetBeans la création de la JavaDoc HTML se fait automatiquement à partir d’un clic droit sur le dossier source, puis « Générer JavaDoc ».

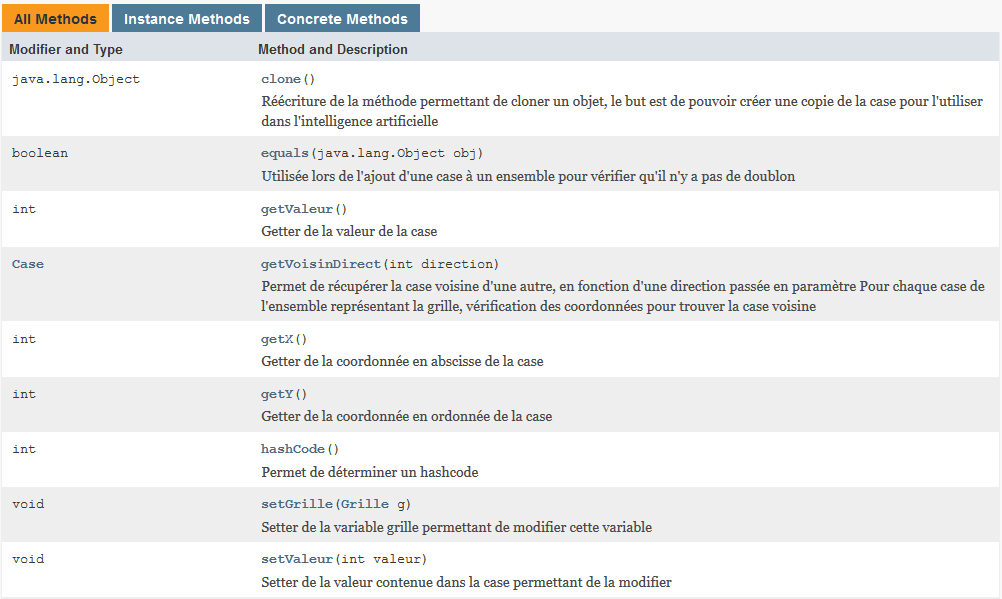
Lorsque nous avons commenté chacune de nos classes et de nos méthodes, nous avons utilisé plusieurs tags en priorité :

* @param, afin de préciser les paramètres de notre méthode ou fonction ou constructeur, il fallait expliquer l’utilité des paramètres pour améliorer la compréhension de l’utilisateur.
* @return documente la valeur de retour. Ce tag n’est évidemment pas utiliser pour les constructeurs ou pour les méthodes ayant pour type de retour *void*.
* @see permet de documenter une association avec une autre méthode ou classe.

Il s’agit des tags basiques, mais nous n’avions pas nécessairement l’utilité des autres. En effet, nous n’avions pas d’exception à traiter, ni de versions différentes, entre autres.

Il ne faut pas oublier que les commentaires sont essentiels dans la conception d’un projet, non seulement pour les éventuels collaborateurs qui pourraient nous venir en aide, mais aussi pour nous si nous souhaitons compléter notre code dans quelque temps.

Voici des aperçus de la documentation que nous avons réalisée :



# RETROSPECTIVE ET CONCLUSION

Pour conclure, nous allons établir un retour sur le travail que nous avons fourni et sur ce que nous pourrions améliorer dans le cas où nous reprendrions ce projet.

D’un point de vue général, nous avons réussi à accomplir une bonne partie des livrables demandés pour ce projet. Ce n’était pas forcément facile, non seulement parce que cela faisait plusieurs mois que nous n’avions pas programmé en Java et d’autres part car les délais étaient limités aux vues des projets que nous avions aussi en parallèle. Ce qui nous a permis de nous débrouiller est le fait que ce projet ait été présenté dès le début du semestre et nous avons su rapidement ce que nous devions rendre précisément. Le fait que le sujet soit la réalisation d’un jeu que nous connaissons bien, nous aide à cibler les besoins plus rapidement.

Aux termes de ce projet, les points les plus sombres sont représentés par le problème connu par l’intelligence artificielle jouant une partie complète sans afficher l’intégralité des déplacements, mais en affichant directement le résultat et l’absence de bouton permettant l’interruption dans celle-ci. Mais aussi, l’absence de Threads dans notre interface graphique. Et enfin, l’impossibilité de jouer en mode multi-joueurs.

Cependant, notre application permet une interface graphique basique, la sauvegarde d’une partie en cours, la consultation du classement des joueurs et l’aide d’une intelligence artificielle pour le prochain coup.

Ce projet nous a permis :

* De revoir la programmation Java, même si ce n’est pas forcément un point fort pour l’ensemble des membres du groupe.
* De découvrir la conception des diagrammes UML.
* D’apprendre à réaliser une interface graphique à l’aide de JavaFX Scene Builder.