Rapport Kakuro : JAVA FX

ZARIKIAN Hayk et DUBERT Alexandre

Mai 2022

1 Introduction

Ce rapport est dédié à notre projet de Programmation orientée objet 2 dans le cadre de notre deuxième année de licence d'Informatique à l'UFR de Mathématique et d'Informatique à l'Université de Strasbourg. L'objectif de ce projet était de créer un jeu Kakuro et d'appréhender la programmation objet en utilisant une interface graphique. Ce projet, réalisé en langage JAVA avec la bibliothèque graphique JavaFx doit permettre l'affichage d'une fenêtre avec la possibilité de naviguer dans le menu de manière interactive entres les fonctionnalités présente dans celle-ci et de pouvoir jouer au jeu Kakuro. Nous avons choisi de travailler en binôme. Pour ce faire, nous avons utilisé un répertoire GIT sur le GitLab de l'université. Nous nous somme réparti les tâches au préalable, et dès que nous finissions une étape importante, nous l'envoyions sur le dépôt distant.

Les rôles ont été réparti de la manière suivante :
- ZARIKIAN Hayk : Développement orienté graphique
- DUBERT Alexandre : Développement orienté jeu

2 Diagramme UML

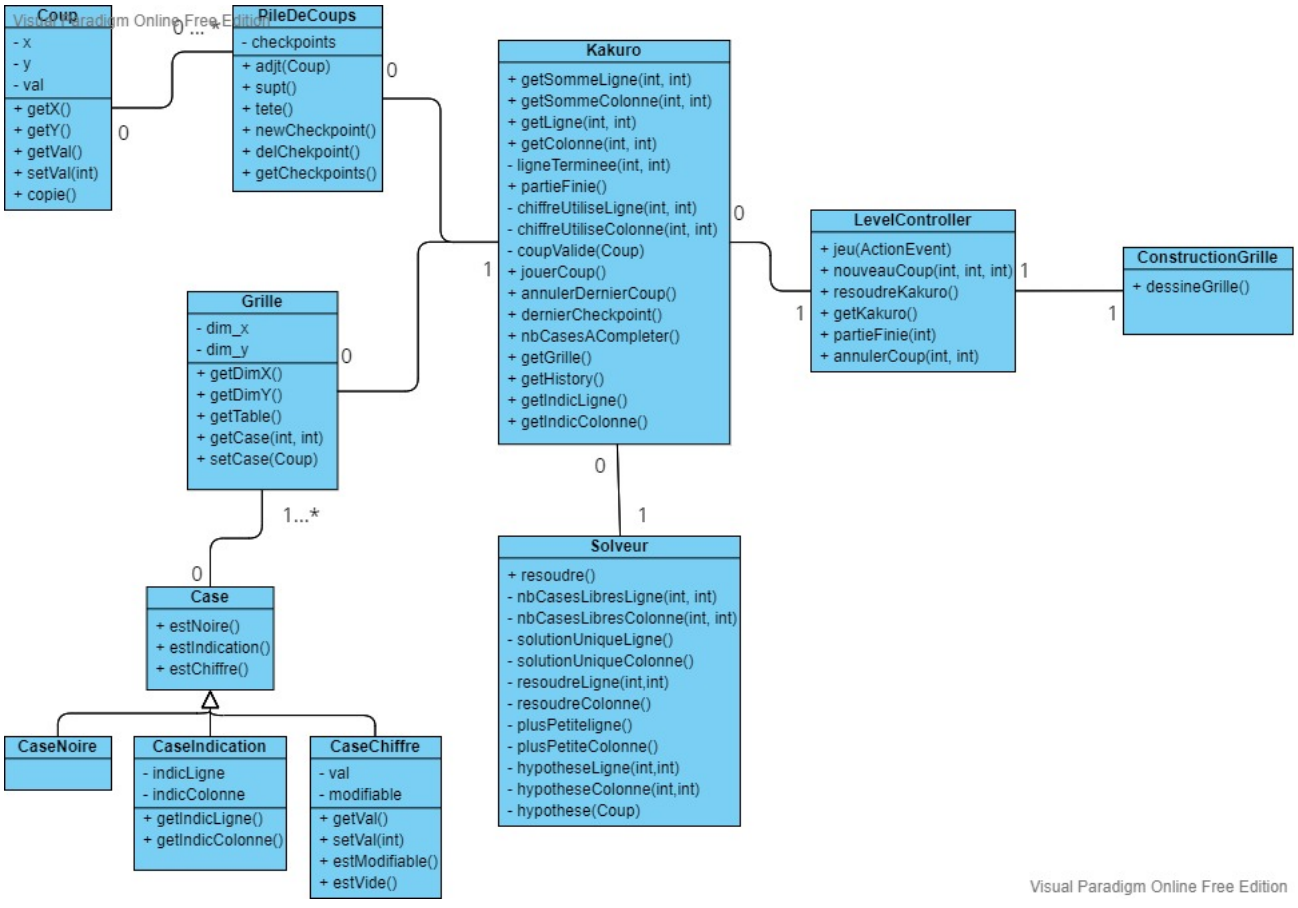

Figure 1: Diagramme UML

3 Le jeu

Tout d'abord, une case peut être une CaseIndication, CaseNoire ou CaseChiffre. CaseNoire n'a aucun attribut, CaseIndication a une indication horizontale et une indication verticale, enfin CaseChiffre a une valeur entière et un attribut pour permettre la modification ou non.
Ensuite une grille est un tableau à deux dimensions de cases. La grille est construite à l'aide d'un niveau. La première ligne du niveau indique la dimension de la grille. Les lignes suivantes sont les lignes de la grille. 0 indique une case noire, a,b indique une case d'indication, un chiffre une case de chiffre non modifiable et un tiret du bas une case de chiffre modifiable. Le kakuro est composé d'une grille, nous avons d'abord voulu le faire hériter de grille, car un kakuro est une grille à laquelle on donne des règles. Mais nous avons finalement privilégié la composition car plus simple pour ce dont nous avions besoin. Dans la plupart des méthodes, on passe en argument les coordonnées d'une case et la méthode renvoie une information sur cette case (indication, ligne, somme de la colonne, etc). Une ligne/colonne au sens du kakuro commence par une indication et se termine soit par une case noire, soit par une nouvelle indication ou soit par la fin de la grille. Donc dans chaque parcours de ligne/colonne, d'abord on rembobine jusqu'à la dernière indication et puis on parcourt jusqu'à la fin ou la prochaine indication/noire. Le deuxième attribut de Kakuro est un historique des coups. Il sert à annuler un coup et il servira au solveur à revenir au dernier checkpoint.
Après il y a la pile de coups. PileDeCoups est une ArrayList de coup, qui a une liste de checkpoint. La classe implémente les méthodes basiques d'opérations sur les piles, avec en plus la création d'un nouveau checkpoint. Le nouveau checkpoint correspond à l'indice actuel de la pile de coups.
Enfin le solveur qui va résoudre le kakuro. Le solveur a un kakuro. La résolution se fait à l'aide de sa fonction résoudre, qui renvoie la grille de son kakuro complétée. L'algorithme est le suivant : S'il y a une incohérence on revient au dernier checkpoint et on retente le coup avec une plus petite valeur, si aucune valeur n'est possible, alors on supprime le checkpoint et on revient au checkpoint précédent. Sinon si une ligne ou une colonne n'a plus qu'un seul élément a remplir alors on résout cette ligne/colonne. Sinon on va faire une hypothèse. L'hypothèse se fait sur la ligne ou la colonne où il y a le moins d'inconnues pour une meilleure probalitité d'avoir juste. Ensuite dans la ligne/colonne, on choisit la case où il y a l'indication la plus élevée, on joue le coup le plus haut possible et on crée un checkpoint dans l'historique des coups. On répète cette boucle jusqu'à ce que la partie soit terminée.

Pour lier les données et l'interface nous avons adopté une architecture MVC. Le contrôleur a la vue et la vue a le contrôleur Le contrôleur intialise le kakuro et la vue. Ensuite si la vue a besoin d'obtenir ou d'envoyer une donnée elle passe par le contrôleur.

4 Interface graphique

4.1 JavaFx

Dans cette première partie concernant l'interface graphique, il est important de parler pourquoi nous avons choisi d'utiliser JavaFx pour notre projet. Tout d'abord, nous avions le choix entres trois interfaces AWT, Swing ou JavaFx pour le langage Java. En prenant en considération qu'AWT et Swing se font très anciens et semble plus restrictif alors notre choix s'est porté sur JavaFx qui propose des possibilités bien plus moderne. Mise à part le langage Java nous avons couplé le langage visuel XML (FXML sur JavaFx) et le langage CSS ce qui à rendu notre jeu plus esthétique.

4.2 Launcher

Le launcher est la base même de notre jeu car c'est grâce au launcher que nous allons pouvoir naviguer dans le menu. Pour ce faire, nous avons utilisé des fichiers FXML et des controllers pour la navigation du menu. Chaque bouton est assigné à une action qui renvoi vers une section.

Dans le launcher se trouve :

- PLAY -> Choix de la grille - TUTORIEL - CLASSEMENT - OPTIONS

Lorsque nous cliquons sur le bouton PLAY, nous arrivons dans la sélection de la grille, nous avons la possibilité d'enregistrer un nom (pas obligatoire) qui va permettre d'inscrire le joueur dans le tableau des scores. Puis, nous avons le choix entres 3 niveaux avec le niveau de difficulté, par défaut il sera sur facile. Le niveau normal et difficile actionne un compteur qui va rendre le

temps de la résolution de la grille limité.

4.3 Fenêtre de jeu

La fenêtre de jeu est construite via la classe ConstructionGrille (pas de FXML) qui est crée lors de la sélection du niveau par des fichiers textes qui sont nos niveaux. Cette fenêtre est dimensionnée par défaut selon l'écran de l'utilisateur ce qui permet de jouer sur différente taille d'écran. Par la suite, ConstructionGrille fait appel aux méthodes qui permettent de dessiner la grille, on utilise la classe ConstructionPolygon qui dessine les rectangles ou triangles pour les indications. Chaque case est définie selon la taille de l'écran car nous allons diviser la taille de l'écran en X par le nombre de case en X, respectivement pour la hauteur. Les cases sont dessinées en conséquence avec l'ajout des zones de textes qui sont éditable par l'utilisateur pour insérer une valeur.

5 Fonctionnalités

5.1 Fonctionnalité 1 : Tutoriel

L'une de nos fonctionnalités est l'accès à un tutoriel via le menu du launcher. Ce tutoriel permet d'expliquer les règles du jeu de manière interactive. Pour ce faire, nous avons utilisé un fichier FXML accompagné de son controller assigné. Le but étant de cliquer sur un bouton qui bascule vers une nouvelle explication avec un déplacer des images ou bien le changement de texte. Le controller fonctionne d'une manière assez simple, il commence par une méthode de départ qui par la suite en appuyant sur le bouton, il récupère un événement assigné au bouton et exécute l'action. Cependant, ne voulant pas créer plusieurs objets de manière redondante ou plusieurs fichiers FXML, nous avons eu l'idée de déplacer les objets et de changer l'action du même bouton suivant là où nous nous trouvons dans le tutoriel.

5.2 Fonctionnalité 2 : Classement

La deuxième fonctionnalité est la mise en place d'un classement des scores. D'une part un score est un nom et une valeur. Le nom doit être unique dans le classement donc la méthode miseAJour compare l'objet actuel avec un autre score et le remplace si le score est plus élevé. D'autre part le classement a une liste de score et un fichier. Sa liste de score est initialisée à partir du fichier. Ensuite on peut ajouter un score et lorsque la partie se termine mettre à jour le classement dans le fichier.

5.3 Fonctionnalité Annexe

Mise à part, les deux fonctionnalités majeures, notre jeu possède des petites fonctionnalités très pratique pour une utilisation interactive.

- Musique de démarrage style japonaise pour la cohérence avec le jeu.
- Possibilité de régler le volume via le menu, ce réglage est enregistré dans toute les sections de l'interface.
- Un son lors du passage de la souris sur un bouton.
- Dans OPTIONS : changement de la taille de la fenêtre du jeu et désactiver le son sur le passage de la souris sur un bouton.
- Gestion d'erreur au niveau de la saisie d'une valeur dans une case.

Il y a bien d'autres fonctionnalités que nous laisserons les utilisateurs découvrir en jouant.

6 Conclusion

Ce projet nous a beaucoup apporté sur deux plans. Premièrement sur le plan de la pratique dans la programmation orientée objet et la manipulation de l'interface graphique. En effet, il a fallut s'approprier une nouvelle interface graphique JavaFx, la comprendre, découvrir les possibilités qu'elle nous offres. Et deuxièmement sur le plan du travail d'équipe. Le partage du travail était très équilibré, la communication très bonne. Nous nous somme mutuellement donnés des idées et conseillés. L'utilisation d'un dépôt GIT était une bonne idée et nous a fait gagner en efficacité. Finalement ce projet a été très plaisant à mener à bien et riche, le résultat est très satisfaisant et nous avons beaucoup progressé.