Rapport – Projet SAé de Java : Sudoku

1 Table des matières

2 Introduction

Le présent rapport documente le projet SAé de Java, qui vise à réaliser deux programmes pour un Sudoku. L'un des programmes consiste à offrir au joueur un moyen de créer et de personnaliser des grilles de Sudoku. Et un deuxième qui permet au joueur de jouer au Sudoku et de trouver une solution pour une grille.

3 Fonctionnalités des Programmes

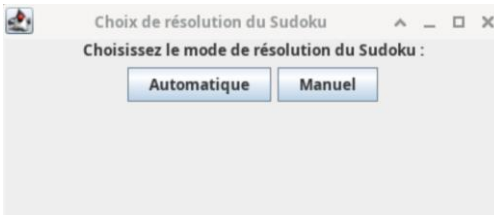
3.1 Programme de Jeu

3.1.1 Main.java

La méthode Main se trouve dans le fichier MainJeu.java, c'est lui qui est exécuter. A l'exécution la méthode afficher()de l'objet FenetreChoix est appelée pour afficher la fenêtre qui permettra au joueur de choisir entre le mode manuel et le mode automatique.

3.1.2 FenetreChoix.java

FenetreChoix affiche donc une fenêtre avec deux boutons (automatique et manuel) qui respectivement sont une instance de AutomatiqueActionListener et ManuelActionListener.



3.1.3 AutomatiqueActionListener.java

Dans AutomatiqueActionListener.java, ce trouve une méthode qui attend que le joueur clique sur le bouton Automatique de FenetreChoix. FenetreChoix est après disposer pour laisser place au joueur la selection de fichier avec JFileChooser qui est ensuite lus par Lecteur.java (voir 3.2 Programme de Création). Pour finir la méthode de Resolveur.java est utilisé pour compléter la grille Sudoku avec le temps mis en millisecondes avec nanoTime. Le résultat de la grille est affiché avec FenetrePrincipale.

3.1.4 Resolveur.java

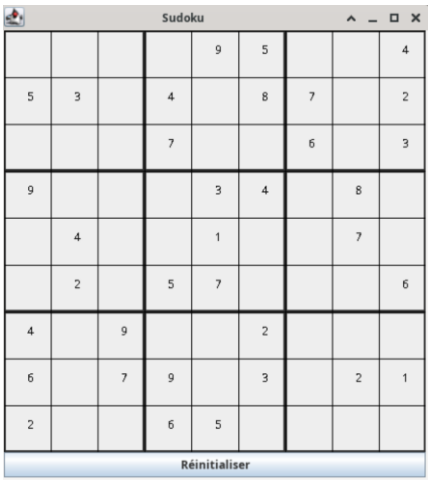
Resolveur.java contient l'algorithme de résolution de Sudoku (voir 5 Algorithme de résolution)

3.1.5 ManuelActionListener.java

Comme pour AutomatiqueActionListener, une méthode attend que le joueur clique sur le bouton Manuel de FenetreChoix. FenetreChoix est après disposer pour laisser place au joueur la sélection de fichier avec JFileChooser qui est ensuite lus par Lecteur.java (voir 3.2 Programme de Création). La grille est ensuite affiche en appelant une méthode de FenetrePrincipale.java.

3.1.6 FenetrePrincipale.java

Une méthode afficher(boolean modeAutomatique, int[][] grille) permet de savoir si c'est le mode automatique ou manuel. Il y a une instance de classe SaisieGrille, qui permet au joueur de saisir des valeurs. Si c'est en mode manuel un bouton Réinitialiser est afficher. Et enfin quelque option pour la taille et la position de la fenêtre.



3.1.7 Reinitialiser.java et ReinitialiserActionListener.java

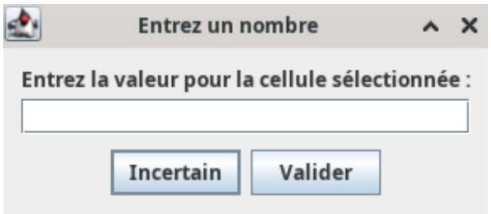
S'occupe de remettre la grille à zéro en supprimant toutes les valeurs entrer par le joueur.

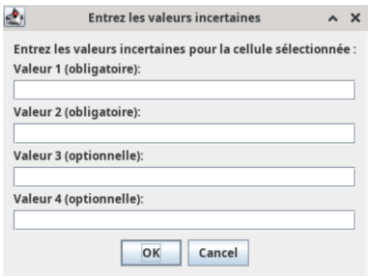
3.1.8 SaisieGrille.java

Au début il y a des attributs qui permettent de récupérer les tailles, les valeurs par défaut et la case que le joueur sélectionne. Ensuite le constructeur qui initialise les attributs. Et pour finir plusieurs méthodes qui permet de vérifier la case que le joueur sélectionne, de valider les nombres entrer et de vérifier que le joueur à terminer la grille.

3.1.9 CaseMouseListener.java

C'est la boite de dialogue qui permet au joueur de rentrer des valeurs. Il peut soit rentrer une valeur unique, ou des valeurs incertaines. Le joueur doit donc au minium rentrer deux valeurs incertaines et la case est séparer en 4 (nombre de valeurs incertaines maximum) et affiche ces 4 valeurs incertaines.





3.1.10 GrillePainter.java

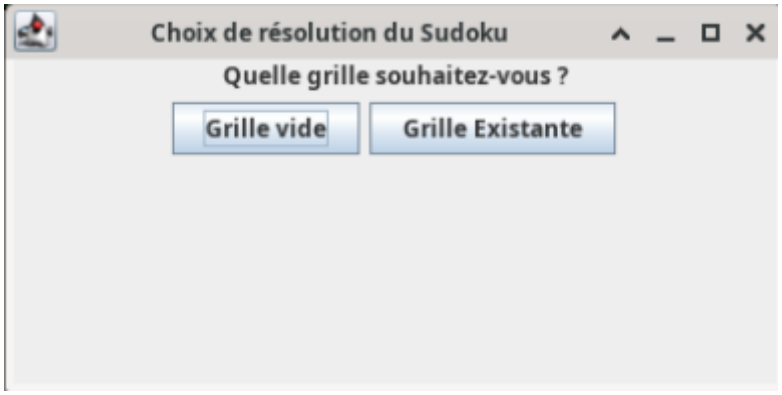
Cette classe permet de dessiner la grille, les régions et les cases. Il sépare les cases en 4 s'il y a besoin pour les valeurs incertaines.

3.2 Programme de Création

3.2.1 MainCreation.java

MainCréation.java, permet d'exécuter le programme. Il lance une fenêtre avec les choix suivants :

- Grille vide, pour créer une grille vide
- Grille Existe, qui permet de lire une grille qui existe déjà



Dans les deux cas, une fenêtre est ouverte avec le choix que l'on a sélectionné.

3.2.2 GrilleExisteActionListener.java

Permet de lancer la fenêtre de sélection de fichier et le programme Lecteur.java. Lance directement FenetreModif.java pour voir la grille à la fin. Son action se réalise quand on appuie sur le bouton.

3.2.3 GrilleVideActionListener.java

Permet la création d'une grilles vide. Elle utilise Grille.java pour initialiser la grille. Lance directement FenetreModif.java pour voir la grille. Son action se réalise quand on appuie sur le bouton.

3.2.4 FenetreModif.java

Ouvre une fenêtre, avec l'affichage de la grille avec SaisieGrille.java (vu plus haut). Elle possède aussi un bouton « Enregistrer », afin de sauvegarder ses documents. Lance une fenêtre de sélection de fichier afin de pouvoir sélectionner un fichier ou créer un fichier pour sauvegarder les données de la grille.

3.2.5 Register.java

Permet d'enregistrer les données dans le fichier sélectionnée, en mettant les données sous un format différent de int afin d'être accepté pour les fichier en .gri, et pouvoir etre relu par le programme Lecteur.java.

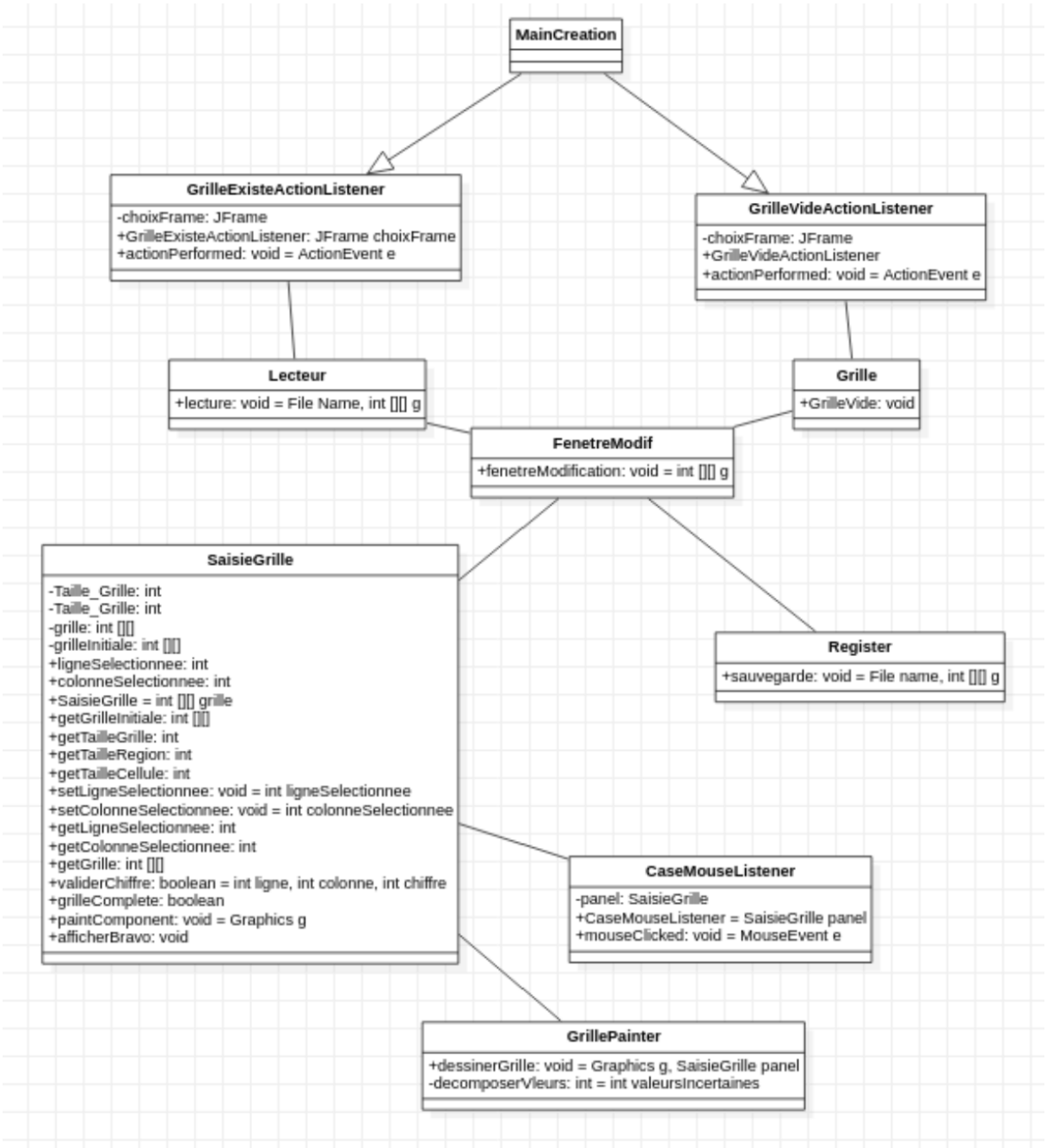
3.2.6 Grille.java

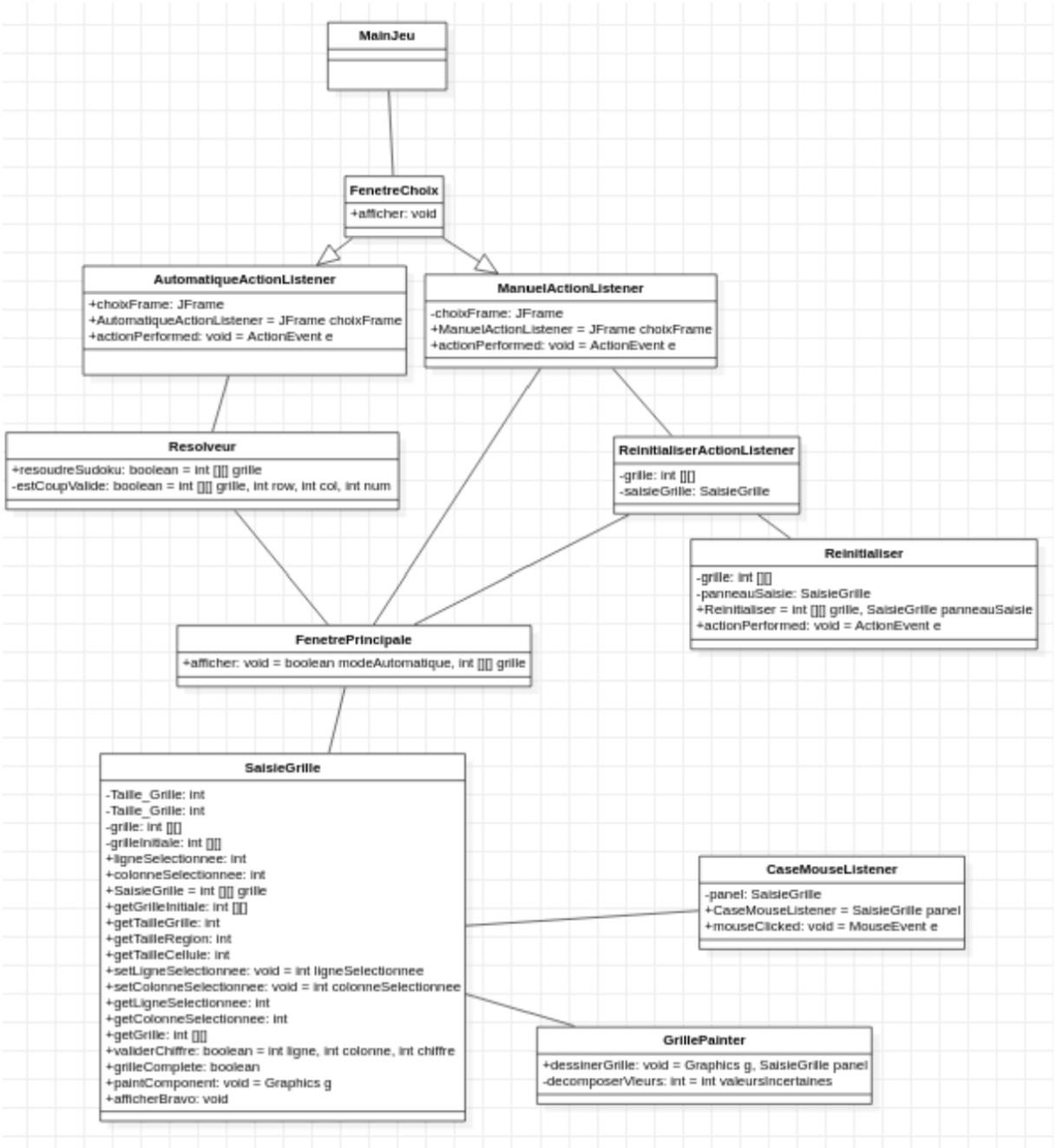
Permet d'initialiser une grille vide. Avec le type void, elle ne renvoie rien mais peut remplir une grille, quand il est mis avec d'autres programme.

3.2.7 Lecteur.java

Permet de lire un fichier et de déchiffrer les données contenues à l'intérieur. Avec le type void, elle ne renvoie rien mais peut remplir une grille, quand il est mis avec d'autres programme.

4 Structure des Programmes




MainJeu
FenetreChoix
+afficher: void
AutomatiqueActionListener
+choixFrame: JFrame
+AutomatiqueActionListener = JFrame choixFrame
+actionPerformed: void = ActionEvent e
ManuelActionListener
-choixFrame: JFrame
+ManuelActionListener = JFrame choixFrame
+actionPerformed: void = ActionEvent e
Resolveur
+resoudreSudoku: boolean = int [][] grille
-estCoupValide: boolean = int [][] grille, int row, int col, int num
ReinitialiserActionListener
-grille: int [][]
-saisieGrille: SaisieGrille
Reinitialiser
-grille: int [][]
-panneauSaisie: SaisieGrille
+Reinitialiser = int [][] grille, SaisieGrille panneauSaisie
+actionPerformed: void = ActionEvent e
FenetrePrincipale
+afficher: void = boolean modeAutomatique, int [][] grille
SaisieGrille
-Taille_Grille: int
-Taille_Grille: int
-grille: int [][]
-grilleInitiale: int [][]
+ligneSelectionnee: int
+colonneSelectionnee: int
+SaisieGrille = int [][] grille
+getGrilleInitiale: int [][]
+getTailleGrille: int
+getTailleRegion: int
+getTailleCellule: int
+setLigneSelectionnee: void = int ligneSelectionnee
+setColonneSelectionnee: void = int colonneSelectionnee
+getLigneSelectionnee: int
+getColonneSelectionnee: int
+getGrille: int [][]
+validerChiffre: boolean = int ligne, int colonne, int chiffre
+grilleComplete: boolean
+paintComponent: void = Graphics g
+afficherBravo: void
CaseMouseListener
-panel: SaisieGrille
+CaseMouseListener = SaisieGrille panel
+mouseClicked: void = MouseEvent e
GrillePainter
+dessinerGrille: void = Graphics g, SaisieGrille panel
-decomposerVleurs: int = int valeursIncertaines

5 Algorithme de Résolution

Pour résoudre notre Sudoku, nous avons choisi d'utiliser un algorithme de **backtracking**, car il est facile à implémenter et bien que le temps de calcul puisse être long avec une grande taille de grille, dans notre cas de 9x9 ce n'est pas vraiment problématique.

5.1 Définition

Qu'est-ce que le **Backtracking** ? Le **backtracking** explore de manière systématique toutes les possibilités d'une solution, en faisant des choix à chaque étape, validant ces choix, et faisant marche arrière en cas d'impasse pour explorer d'autres options.

5.2 L'algorithme

1 - Recherche d'une cellule vide : L'algorithme commence par rechercher une cellule vide dans la grille Sudoku. Si aucune cellule vide n'est trouvée, cela signifie que la grille est complète et donc résolue.

2 - Tentative de placement des chiffres : Une fois une cellule vide trouvée, l'algorithme essaie de placer les chiffres de 1 à 9 dans cette cellule, un par un.

3 - Validation des coups : Avant de placer un chiffre dans une cellule vide, l'algorithme vérifie si ce coup est valide selon les règles du Sudoku. Cela implique de vérifier que le chiffre n'est pas déjà présent dans la même ligne, la même colonne ou la même région 3x3.

4 - Résolution récursive : Si un coup est valide, l'algorithme récursivement tente de résoudre le reste de la grille en appelant la fonction resoudreSudoku sur la grille modifiée. Si cette tentative de résolution est réussie, cela signifie que la grille est résolue et l'algorithme renvoie vrai.

5 - **Backtracking** : Si la tentative de résolution échoue, l'algorithme annule la valeur placée dans la cellule et continue à explorer d'autres possibilités en plaçant d'autres chiffres dans la cellule vide. Cela est possible grâce à la nature récursive de l'algorithme, qui permet de revenir en arrière (backtrack) et d'explorer d'autres chemins si nécessaire.

6 - Retour false : Si aucune solution n'est possible pour la cellule vide actuelle, l'algorithme renvoie faux, ce qui signifie qu'il doit y avoir une erreur dans les choix précédents. Cela déclenche le backtrack pour explorer d'autres options dans les cellules précédentes.

6 Conclusion Personnelle

Adrien Dick – Ce projet en groupe en Java a été enrichissante car cela a permis, de combiner nos compétences individuelles pour créer quelque chose de fonctionnel. De développer mes compétences en programmation Java. Jai pu aussi me performer dans l'utilisation de git. Mais aussi de Word pour essayer de fournir un rapport le plus qualitatif possible.

Hugo Raban – Ce projet nous a permis d'utiliser des moyens d'interagir avec le système. Cela m'a fait découvrir des fonctions et aussi d'améliorer mes compétences en JavaScript. Mais également de pouvoir mieux utiliser les diagrammes pour un cas pratique, afin de réaliser le programme correctement. Ainsi que de pouvoir expliquer en détails le fonctionnement du programme.

7 Annexes

Aide de ChatGPT à la rédaction du rapport pour l'orthographe et la syntaxe

Extrait du rapport original (avant correction par ChatGPT) :

"Le présent rapport documente le projet SAé de Java qui est de réaliser deux programmes pour un Sudoku. L'un des programme consistent à donner au joueur un moyen de créer et personnaliser des grille de Sudoku. Et un deuxième qui permet au joueur de jouer au Sudoku et trouver une solution d'une grille."

Extrait corrigé après l'intervention de ChatGPT :

"Le présent rapport documente le projet SAé de Java, qui vise à réaliser deux programmes pour un Sudoku. L'un des programmes consiste à offrir au joueur un moyen de créer et de personnaliser des grilles de Sudoku. Et un deuxième qui permet au joueur de jouer au Sudoku et de trouver une solution pour une grille."

Wikipedia. (Consulté le 05/04/24). "Algorithmes de résolution des sudokus." En ligne : https://fr.wikipedia.org/wiki/Algorithmes_de_résolution_des_sudokus.