Rapport du projet de Software Engineering
Projet SudokuSolver

Maxime CELESTE 22210379, Calvin ZEGBEU 22108064

Novembre 2024

Sommaire

1 Introduction

L'objectif de ce projet est de développer un solver de sudoku en Java qui résout les grilles de sudoku en y appliquant un ensemble de règles de déduction et qui juge leur difficulté.

1.1 Création de la grille sudoku

Pour créer la grille du sudoku nous prenons en entrée un fichier texte contenant une grille au format suivant:

- 9 nombres par ligne, de valeur 0 si la case est vide sinon entre 1 et 9.
- les nombres sont séparés par des virgules sur chaque ligne. et nous la linéarisons dans un tableau de taille 8 (nombre de cases dans un sudoku 9*9).

Ensuite on crée un tableau de la classe Choix contenant les listes des valeurs possibles ainsi que le nombre de valeurs possibles pour chaque case.

1.2 Parcours de la grille

Pour pouvoir parcourir la grille dans les règles de déduction nous avons créé des fonctions qui renvoient les tableaux d'indices des différentes zones du sudoku: les 9 lignes, colonnes et boîtes (carrées 3*3), sinon nous parcourons le tableau du début à la fin de façon linéaire.

1.3 Fonctionnement du solver

L'algorithme de résolution utilise trois règles de déduction de la façon suivante: si une règle n'est pas en mesure de résoudre la grille, on passe à la suivante.

En effet, chaque règle équivaut à un niveau de difficulté (respectivement facile, moyenne et difficile), si jamais les règles ne suffisent pas à résoudre la grille, l'utilisateur devra entrer une case dans la grille pour l'aider, à ce moment-là, la grille sera considérée comme "très difficile".

2 Règles de déduction

Pour pouvoir progresser ou régresser dans l'algorithme, nous avons fait en sorte que chaque règle de déduction renvoie différentes valeurs, elles retournent toutes 0 si elles n'ont pas fonctionné et 1 dans le cas contraire. En cas de 0 comme valeur de retour, on passe à la règle de déduction suivante ou à l'entrée manuelle selon la règle d'où vient le 0, sinon on retourne à la DR1.

2.1 Deduction Rule 1

Notre première règle, DR1 est basée sur la solution "Single candidate" qui se présente de la manière suivante: si il n'y a qu'un seul choix possible pour une case, on lui affecte la valeur de cet unique choix et on la retire de tous les choix de zone concernée par la case, c'est-à-dire la colonne, la ligne et la boîte auxquelles elle appartient.

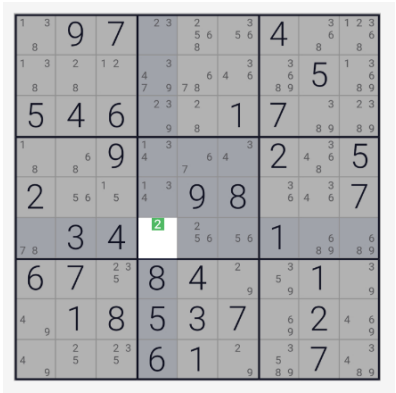

Illustration de la case à choix unique *

2.2 Deduction Rule 2

Notre seconde règle, DR2 est aussi basée sur la solution "Single candidate" qui se présente cette fois de la manière suivante: si un nombre n'apparaît qu'une fois dans une ligne, une colonne ou bien une boîte, on affecte la valeur en question
à la case associée à ce choix et on retire la valeur des choix possibles de la zone concernée par ladite case.

Illustration de l'apparition unique du choix 3*

2.3 Deduction Rule 3

Notre troisième règle, DR3 se base quant à elle sur la technique "Naked tuple", une généralisation des "Naked pair" et "Naked quad" qui consiste à vérifier si on peut avoir un choix différents répartis sur n cases dans une zone prédéfinie (ligne, colonne, boîte) et à les retirer des choix du reste de la zone. Pour chaque case vide C avec n choix dans une zone, on cherche à savoir si les choix des autres cases sont inclus dans les siens, pour ce faire on compare les autres cases avec les n sous-ensembles de ses choix de taille n-1 ou avec son unique sous-ensemble de taille n et on les compte. Si le compteur est égal à n et qu'il y a plus de n cases vides dans la zone, alors on retire les choix de C à chacune des cases vides.

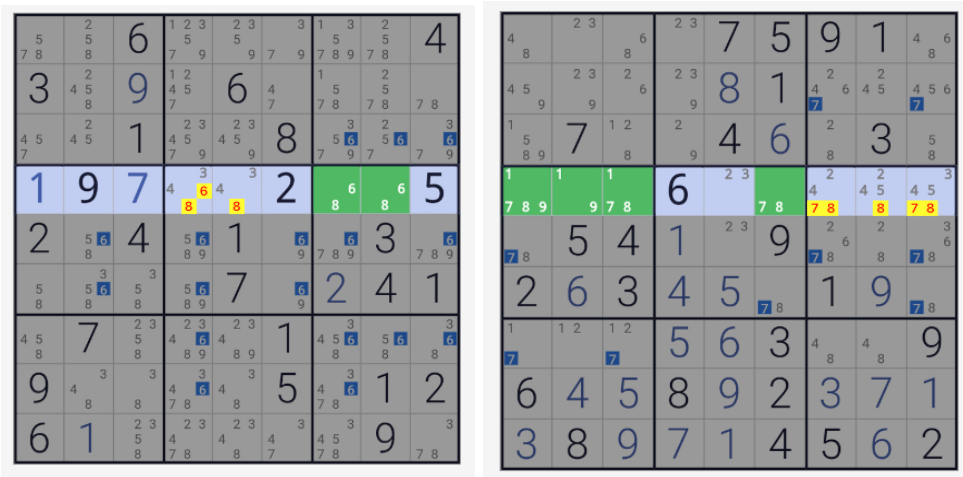

illustration du naked pair (à gauche) et du naked quad (à droite)*

- illustrations venant de l'application Sudoku de Guru Puzzle games

3 Design Patterns utilisés

L'une des contraintes du projet est qu'il fallait utiliser au moins 4 design patterns (ou patrons de conception) et nous avons choisi les design patterns suivants:

3.1 Singleton

Nous avons utilisé un Singleton afin de n'avoir qu'une seule instance de type Grille créée pour le fonctionnement du Solver. Cela permet l'accès unique à un seul type de Grille durant tout le déroulé de l'algorithme.

3.2 Strategy

Nous avons utilisé une Strategy pour l'utilisation des Deduction Rules qui nous permet d'utiliser la bonne règle de déduction selon le contexte donné dans le main.
Chaque objet contient une référence vers une Deduction Rule.

3.3 Memento

Nous avons utilisé le Mémento comme design pattern afin de stocker la grille générée au tout début du programme ainsi que le tableau de choix au cas où il y aurait un problème avec une des entrées manuelles nous forçant à redémarrer le programme de 0 à l'exception de la difficulté de la grille qui reste évidemment inchangée

3.4 Observer

Nous avons utilisé l'Observer comme design pattern afin d'avoir un suivi des valeurs dans la grille lors de son initialisation, lors de l'affectation de nouvelles valeurs ainsi que la fin de l'exécution du programme.

4 Conclusion

Nous avons réussi à répondre aux objectifs du projet avec une solution en programmation orientée objet en java pour prendre une grille en entrée et la résoudre tout en jugeant sa difficulté en utilisant les principes vus en classe tout en respectant l'énoncé..