Université De Lille
Faculté des Sciences et Technologies
Département de Mathématiques

Compte rendu du projet programmation avancée

Réalisé par :

- Abdelmonssif *Oufaska*
- Kamal *Samnouni*

Enseignant :

- *Prof.* Yvan *Stroppa*

Année Universitaire :
2020/2021
26 Janvier 2021

Introduction

Le sudoku est un jeu présenté sous la forme d'un carré de 81 cases, le principe du jeu est de remplir le carré selon différentes contraintes. Le but de ce projet est de créer une interface graphique en Java pour le sudoku permettant à un utilisateur de cliquer dans des cases et de simplifier la vision du jeu grâce à des affichages de couleurs. Pour réaliser tout cela, il a tout d'abord fallu créer l'interface graphique. On s'est intéresse au début d'utiliser un solveur d'optimisation linéaire (**choco-solver** et **GLPK**) dans le but de résoudre le jeu du sudoku par la résolution d'un problème d'optimisation linéaire sous contraintes. Mais on a pas réussi à résoudre le problème de dépendance entre les packages de ces solveurs et le package utilisé pour le mode graphique. D'après des recherches sur internet, on a trouver qu'on peut résoudre le jeu du Sudoku par la méthode de **bracktraking**, même si la complexité de cet algorithme pour la résolution est très coûteuse. On va expliqué cette méthode dans la suite de ce projet

Notre interface graphique du jeu est de la forme suivante :

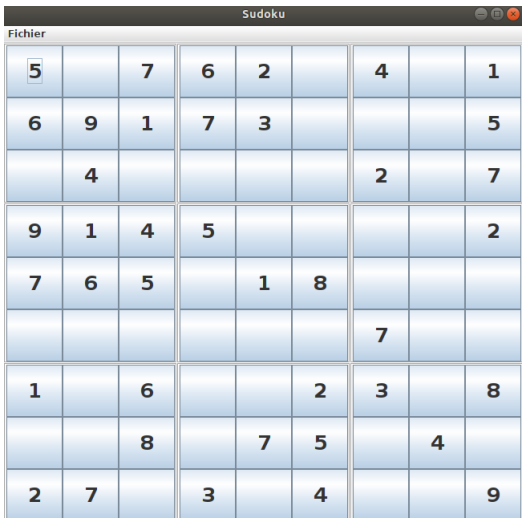


Table des matières

Sudoku

1 Historique du sudoku

Le sudoku est un jeu en forme du carré défini en 1979 par l'Américain Howard Garns, mais inspiré du carré latin, ainsi que du problème des 36 officiers du mathématicien suisse Leonhard Euler. Le but du jeu est de remplir la grille avec une série de chiffres (ou de lettres ou de symboles) tous différents, qui ne se trouvent jamais plus d'une fois sur une même ligne, dans une même colonne ou dans une même grille. La plupart du temps, les symboles sont des chiffres allant de 1 à 9, les grilles étant alors des cases de 3×3. Quelques symboles sont déjà disposés dans la grille, ce qui autorise une résolution progressive du problème complet.

2 Présentation des packages

Dans notre programme on dispose de deux packages, sur le premier on crée trois classes qui permet de faire toutes les opération mathématiques et les vérifications nécessaires. Sur le deuxième on crée la classe principal qui gère le premier package et qui permet l'affichage de mode graphique.

2.1 Package modele

On utilisera dans ce jeu les trois classes suivantes permettent de représenter un sudoku dans la console.

2.1.1 Classe Case

La classe Case représente une case de la grille, elle contient donc une valeur numérique et un état qui définit si la case est fixe ou non, quand c'est fixe on peut pas changer la valeur de la case, de cette façon on peut présenter trois niveaux du jeu, difficile, moyen et

facile. Par exemple, si la grille a plusieurs cases vides donc il s'agit d'un niveau facile.

La classe Case, contient aussi des accesseurs (getFixe) et modifieurs (setFixe) qui permettent aux autres classes pour accéder à ses champs, pour modifier les valeurs si la case n'est pas dans un état fixe. Et aussi une méthode toString pour l'affichage d'une case. On représente ici un aperçu de la classe Case

```
public class Case {
    /***
     * entier représente la valeur de la case
     */
    public int num;
    /***
     * un boolean représente l'état de la case
     *
     */
    public boolean fixe;
    .
    .
    .
    .
    .
}
```

2.1.2 Classe Grille

La classe Grille représente une grille de 3x3 cases. Elle contient des constructeurs, accesseurs et modifieurs comme dans la classe Case, et aussi des méthodes de vérification.

La méthode GrilleComplete() qui vérifie si la grille est complète, donc elle retourne True si elle est composée de tous les chiffres de 1 à 9, et False si non. L'autre méthode GrilleValide(int val) vérifie si une valeur val peut être insérée dans la grille, elle retourne False si la valeur val existe déjà dans la même grille. Et aussi une méthode toString pour l'affichage d'une grille. On représente ici un aperçu de la classe Case

```
public class Grille {
    /***
     * Grille de 3x3 Cases
     */
    public Case[][] _Grille;
    .
    .
    .
    /***
     * Verifié si la grille contient des chiffres
     * de 1 a 9 une fois, retourne vrai ou faux
     */
    public boolean GrilleComplete() {
        .
        .
    }
    /***
     * Verifié si la grille en cours de création
     * serait validé avec la valeur val, return vrai ou faux
     */
    public boolean GrilleValide(int val) {
        .
        .
    }
    .
    .
}
```

2.1.3 Classe Jeu

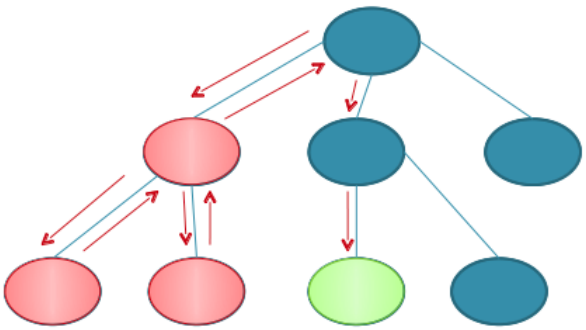
Enfin, la classe Jeu définit un carré de 3x3 grilles. Elle contient un constructeur par défaut pour construire une grille vide. En plus de ce constructeur, elle contient des accesseurs, modifieurs et des méthodes de vérification similaires à celles de la classe Grille mais qui s'appliquent aux lignes et aux colonnes.

Les deux méthodes ligneComplete(int i) et colonneComplete(int j) pour vérifier que chaque ligne i et chaque colonne j contient une valeur une et une seule fois. Et deux autre méthodes ligneValide(int i, int val) et colonneValide(int j, int val) pour vérifier si la valeur val existe une seule fois dans la colonne j et la ligne i, elles retournent True si c'est vrai et False si non. La méthode gagne() vérifie si le joueur a gagné, une fois le joueur a rempli toutes les cases, elle vérifie si les grilles sont bien remplis et si les lignes et les colonnes sont aussi bien remplis, True si le joueur gagne et False sinon.

```
public class Jeu {
    public Grille[][] jeu;
    ...
    public boolean ligneComplete(int i) {
        ...
    }
    public boolean ligneValide(int i, int val) {
        ...
    }
    public boolean colonneComplete(int j) {
        ...
    }
    public boolean colonneValide( int j, int val) {
        ...
    }
    public boolean gagne() {
        ...
    }
    public void RemplissageAleatoire(int s) {
        ...
    }
    public boolean resoudre(int i, int j) {
        ...
    }
}
```

Pour permettre de remplir le carré de façon aléatoire à chaque fois on veut jouer, on utilise pour cela une autre méthode RemplissageAleatoire(int k) qui permet de créer un sudoku de k nombre parmi 81 d'une manière aléatoires. Pour être sûr qu'il existe une solution au jeu, la méthode va résoudre au début un jeu de $s = 13$ nombre aléatoires qui vérifient les contraintes, car on sait qu'il existe toujours une solution au jeu du Sudoku avec moins de 17 nombres fixes. Ensuite on choisit k nombre voulu depuis la solution trouvé dans des positions aléatoires pour s'assurer l'existence d'une solution pour le jeu aléatoire proposé, on les insère dans le carré et on fixe les cases. Ici on fixe les trois niveaux suivants ; si $k = 30$ le niveau est facile, $k = 20$ le niveau est moyenne et si $k = 10$ le niveau est difficile. Pour trouver la solution d'un sudoku d'après un jeu incomplet, on utilise la méthode resoudre c'est une méthode de backtraking :

Le backtracking est une forme de parcours en profondeur d'un arbre avec des contraintes sur les noeuds.
L'idée est de partir du noeud parent, descendre dans le premier noeud fils satisfaisant la contrainte que les grilles, les colonnes et les lignes restent toujours valides. Ce noeud fils devient alors un noeud parent et l'on parcourt ensuite ses noeuds fils sous le même principe.



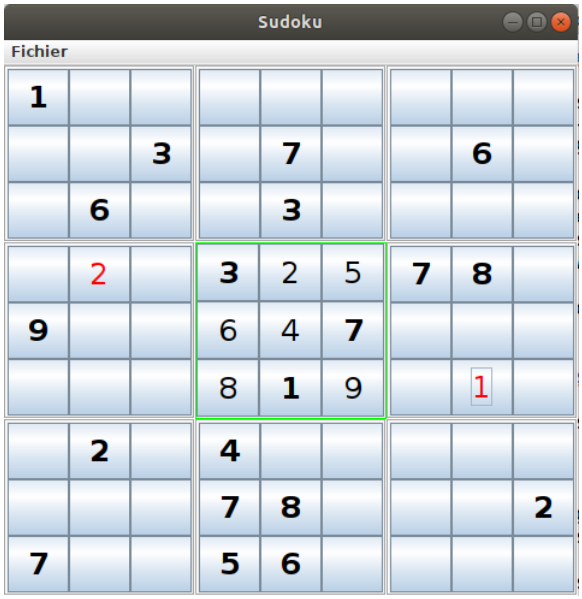
Lorsque l'on a parcouru tous les noeuds fils d'un noeud et qu'aucun ne satisfait la contrainte, on remonte alors au noeud parent et on descend dans le noeud fils suivant. Si l'on arrive au dernier fils du premier noeud parent et qu'il ne satisfait pas la contrainte alors il n'existe pas de solution. La solution est identifiée lorsque l'on arrive à un noeud qui satisfait la contrainte et qui n'a pas de noeud fils.
Enfin une méthode toString afin de pouvoir afficher les résultats de tests de manière claire dans le terminal.

2.2 Package principal

Dans ce package on va créer deux classes Sudoku et Main qui permettent de gérer les trois classes du premier package et de représenter un sudoku dans un interface graphique.

2.2.1 Classe Sudoku

La classe Sudoku pour gérer les trois classes du premier package et de créer un mode graphique pour le jeu Sudoku. elle contient un constructeur par défaut qui permet de créer la fenêtre du jeu, représenté par 3×3 grilles chaque grilles contient 3×3 cases, on trouve aussi dans la fenêtre du jeu la barre des menus qui contient des éléments pour choisir le niveau du jeu souhaité (facile, moyen, difficile), pour remplir le jeu aléatoirement et pour résoudre le jeu.



elle contient aussi des méthodes qui permet de changer les couleur du mode graphique. La fonction CaseFix(int i, int j) qui permet de donner aux nombres des cases fixes un couleur gras. ColorerCaseFausse(int i, int j, int val) qui permet de changer la couleur de la case de coordonnés (i, j) en rouge s'elle vérifie les contraintes. La fonction ColorerGrille(int i, int j) qui donne une couleur vert à la bordure de la grille s'elle contient des chiffre de 1 à 9.
La fonction IncrementeCase(int i, int j) qui incrémente le chiffre de la case quand on appuie dessus. Et aussi la fonction actionPerformed(ActionEvent e) qui gére les actions (click et la barre des menus)

2.2.1 Classe Main

La classe Main pour lancer le jeu dans une fenêtre.

Bibliographie

Dans ce projet on s'est basé sur les sites suivants :

https://fr.wikipedia.org/wiki/Sudoku

http://igm.univ-mlv.fr/~dr/XPOSE2013/sudoku/backtracking.html

https://fr.wikipedia.org/wiki/Retour_sur_trace