SAE2.1

2023

Dimitri SOLAR
Basile LEGRELLE
Groupe 1

Table des matières

Introduction *3*

Description du sujet 3

Les objectifs 3

Différentes étapes du projet 3

Fonctionnalités *4*

Création 4

Joueur 4

Import 4

Le projet dispose d'une fonctionnalité d'import des grilles. Ce système est possible pour le mode création et le mode joueur. On importe des grilles sous le format. gri. 4

Export 4

Vérification 4

Le mode joueur dispose d'un bouton « Vérifier » qui permet au joueur de corriger la complétion de sa grille. 4

Résolution 4

Le mode joueur dispose d'un bouton « Résoudre » qui permet au joueur de compléter sa grille. 4

Modification des grilles importées 4

En mode création l'utilisateur est libre de supprimer les chiffres importées par un fichier au format .gri. Il n'est pas possible pour quelqu'un qui joue de le faire, les cases importées sont alors grisée 4

Remplissage des cellules 5

Timer 5

Découpages des fichiers *6*

ButtonPanel.java 6

Cell.java 6

FileHandler.java 6

GridPanel.java 6

PlayerCell.java 7

PlayerGrid.java 7

Solver.java 7

SudokuFrame.java 7

Processus de résolution d'une grille *9*

Conclusion *12*

Dimitri 12

Basile 12

Introduction

Description du sujet

Ce projet consiste en la réalisation d'un jeu de Sudoku en langage Java dans le cadre de notre premier semestre de première année de BUT Informatique.

Le Sudoku est un jeu de réflexion dont le but est de remplir une 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 région (ou bloc, sous-grille ou "box"). Le jeu se complexifie au fur et à mesure que le joueur progresse, avec des grilles initialement partiellement remplies et le joueur doit compléter les cases vides.

Les objectifs

Nous voulions que notre projet soit intuitif, agréable et éducatif. Nous avons cherché à créer une interface utilisateur claire et une expérience de jeu fluide, avec des réponses immédiates à l'interaction de l'utilisateur pour améliorer l'apprentissage et la satisfaction.

Différentes étapes du projet

1/ Création de la fenêtre graphique
2/ Entrée des chiffres
3/ Export/import des grilles
4/ Modification des grilles
5/ Découpage en plusieurs classes
6/ Création du mode joueur
7/ Résolution des grilles
8/ Ajout du timer
9/ Blocage des cellules importées pour le mode joueur
10/ 4 chiffres par case pour joueur
11/ Factorisation du programme
12/ Mise en place des javadoc
13/ Réalisation du rapport

Fonctionnalités

Création

Le programme dispose d'un mode de création qui permet d'éditer les grilles.
Ce dernier est complètement libre et permet d'ajouter ou de supprimer n'importe quel chiffre (tout en vérifiant les contraintes d'unicité).

Joueur

Le programme dispose d'un mode de jeu qui permet de jouer au Sudoku.
Ce dernier dispose de contraintes qui ne sont pas présentes dans le mode de création.

Import

Le projet dispose d'une fonctionnalité d'import des grilles.
Ce système est possible pour le mode création et le mode joueur.
On importe des grilles sous le format. gri.

Export

Le projet dispose d'une fonctionnalité d'export des grilles.
Ce système est possible pour le mode création et le mode joueur.
On exporte les grilles au format .gri.
Cela permet au joueur de sauvegarder sa progression en créant ou en jouant sur une grille.

Vérification

Le mode joueur dispose d'un bouton « Vérifier » qui permet au joueur de corriger la complétion de sa grille.

Résolution

Le mode joueur dispose d'un bouton « Résoudre » qui permet au joueur de compléter sa grille.

Modification des grilles importées

En mode création l'utilisateur est libre de supprimer les chiffres importées par un fichier au format .gri.
Il n'est pas possible pour quelqu'un qui joue de le faire, les cases importées sont alors grisée

Remplissage des cellules

Le créateur peut mettre le chiffre qu'il veut dans une grille.

Le joueur a lui la possibilité de mettre 4 chiffres dans une cellule afin de se préserver différentes possibilités.

La taille des chiffres se réduit alors afin d'augmenter la visibilité.

Timer

Le mode joueur dispose d'une mesure de temps afin de montrer le temps que met le programme a résoudre une grille.

Découpages des fichiers

ButtonPanel.java

Description : Ce fichier définit un panneau pour les boutons de l'interface utilisateur.

Fonctions :

ButtonPanel(): Constructeur qui configure le layout du panneau.
addButton(String, ActionListener): Ajoute des boutons dynamiquement au panneau avec un texte et un gestionnaire d'événements.

Cell.java

Description : Représente une cellule individuelle dans la grille de Sudoku, étendant JTextField et intégrant KeyListener.

Fonctions :

keyTyped(KeyEvent): Gère la saisie de chiffres dans les cellules et rejette les entrées non valides.
keyPressed(KeyEvent), keyReleased(KeyEvent): Méthodes requises par l'interface mais non utilisées.

FileHandler.java

Description : Gère la sauvegarde et le chargement des états de jeu depuis et vers des fichiers.

Fonctions :

loadGridFromFile(GridPanel, boolean): Charge un état de grille depuis un fichier.
exportGridToFile(GridPanel): Sauvegarde l'état actuel de la grille dans un fichier.

GridPanel.java

Description : Gère l'affichage et la logique de la grille de Sudoku.

Fonctions :

GridPanel(): Constructeur qui initialise la grille avec un layout de grille 9x9.
initGrille(): Initialise la grille en plaçant les cellules.
getCellules(): Retourne la matrice de cellules.
isValidInput(int, int, char): Vérifie si une entrée est valide selon les règles du Sudoku.

PlayerCell.java

Description : Cellule personnalisée pour la grille de Sudoku, avec des fonctions supplémentaires spécifiques au jeu.

Fonctions :

setStartingCell(boolean): Détermine si une cellule est une cellule initiale fixe et la rend non modifiable.
handleKeyTyped(KeyEvent): Gère des entrées spécifiques pour modifier l'affichage et la taille du texte en fonction du contenu.

PlayerGrid.java

Description : Spécialise GridPanel pour le mode joueur, avec des cellules interactives.

Fonctions :

initGrille(): Surcharge la méthode pour utiliser des PlayerCell au lieu de Cell simples.

Solver.java

Description : Implémente l'algorithme de résolution du Sudoku.

Fonctions :

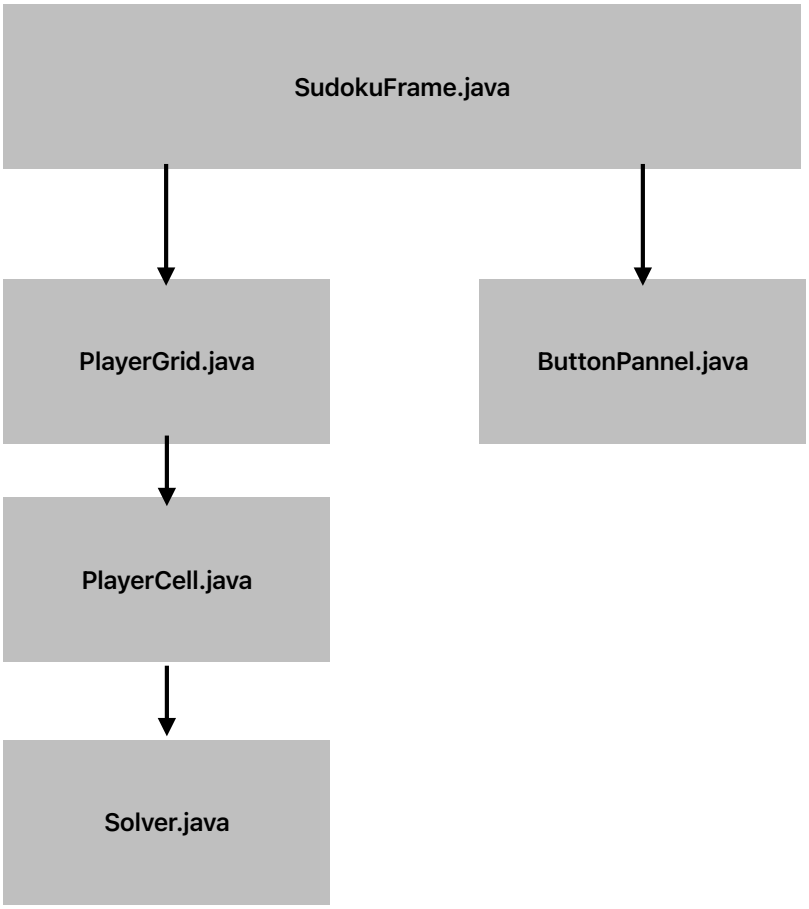
solve(): Tente de résoudre la grille de Sudoku de manière récursive.

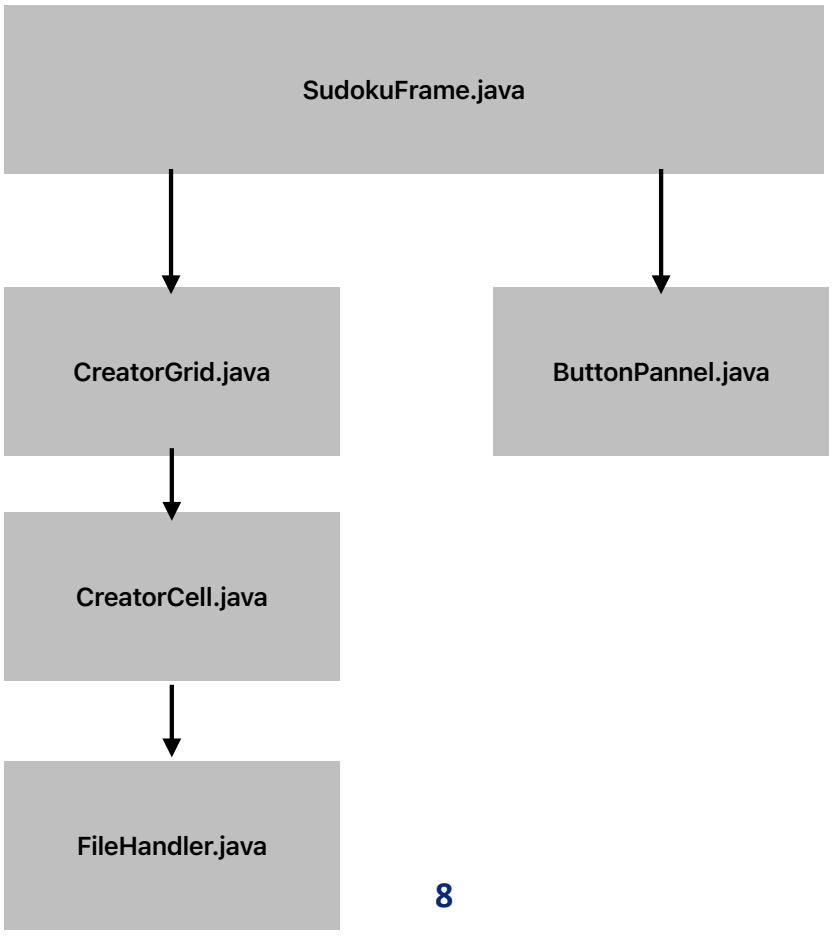
SudokuFrame.java

Description : La fenêtre principale de l'application qui intègre tous les composants.

Fonctions :

SudokuFrame(boolean): Initialise la fenêtre avec un GridPanel et un ButtonPanel.
createButtonPanel(boolean): Configure et ajoute des boutons pour la gestion du jeu.
solveGrid(), verifyGrid(): Fonctions pour résoudre la grille ou vérifier sa validité.


SudokuFrame.java
PlayerGrid.java
ButtonPannel.java
PlayerCell.java
Solver.java


SudokuFrame.java
CreatorGrid.java
ButtonPannel.java
CreatorCell.java
FileHandler.java

Processus de résolution d’une grille

1 Parcours de la Grille :

La fonction solve() parcourt chaque cellule de la grille (GridPanel) du Sudoku, de la première à la dernière, en utilisant deux boucles imbriquées qui itèrent sur les lignes et les colonnes.

2 Vérification des Cellules Vides :

Pour chaque cellule, le programme vérifie si elle est vide (cell.getText().isEmpty()). Une cellule vide est celle qui nécessite l'insertion d'un chiffre.

3 Essai des Chiffres de 1 à 9 :

Si une cellule est vide, la fonction essaie d'y placer chaque chiffre de '1' à '9'. Pour chaque chiffre essayé, le programme vérifie si placer ce chiffre à cet emplacement respecte les règles du Sudoku.

4 Validation de l'Insertion :

La validation est effectuée par la méthode gridPanel.isValidInput(row, col, num). Cette méthode vérifie que le chiffre proposé ne se répète pas dans la même ligne, colonne ou bloc 3x3, conformément aux règles du Sudoku.

5 Récursivité et Backtracking :

Si l'insertion du chiffre est valide, le programme place le chiffre dans la cellule et appelle récursivement la fonction solve(). Si cette nouvelle invocation retourne true, cela signifie que le placement de ce chiffre a conduit à une solution complète et valide, et true est renvoyé pour la récursion en cours.

Si l'insertion du chiffre ne conduit pas à une solution (i.e., la fonction solve() appelée récursivement retourne false), cela signifie qu'une impasse a été atteinte. Le programme efface alors le chiffre de la cellule (cell.setText("")) et essaie le chiffre suivant dans la boucle.

6 Échec à Trouver une Solution :

Si aucun des chiffres de '1' à '9' ne convient pour une cellule donnée (tous ont conduit à false), la fonction solve() retourne false pour sa récursion actuelle, indiquant qu'aucune solution valide n'est possible avec les choix faits précédemment. Cela déclenche le backtracking vers l'état précédent.

7 Achèvement :

La grille est complètement résolue si toutes les cellules sont correctement remplies sans qu'aucun false ne soit retourné par la fonction solve(). Si la fonction atteint le point où toutes les cellules sont traitées et valides, elle retourne true, indiquant que la grille est résolue.

Conclusion

Dimitri

Le projet a été difficile en termes de programmation, en raison des multiples façons de coder pour obtenir le même résultat, ce qui rendait difficile de choisir l'approche la plus optimisée.
La séparation des programmes a été compliquée en raison du nombre de fichiers utilisés et du risque de se perdre ou d'écrire dans le mauvais fichier.
De même, la décision de savoir quelles classes utiliser et comment les séparer était un défi, notamment en déterminant à partir de quel moment il était nécessaire de créer des classes distinctes.
Heureusement, une bonne entente avec mon coéquipier a facilité le bon déroulement du projet.

Basile

C'était la première fois que je me lançais dans un projet Java, et le défi était passionnant. J'ai vraiment apprécié chaque étape du développement, voyant cela plus comme une partie d'échecs que comme un simple devoir.

Ce projet Sudoku, mon deuxième projet de programmation mais le premier en Java, m'a permis de découvrir une nouvelle facette de la programmation. Travailler en Java a été une révélation, notamment en raison de la richesse de cet environnement de développement.

La collaboration en binôme a été particulièrement fructueuse. Partager les tâches, réfléchir ensemble aux solutions et surmonter les défis techniques ont renforcé ma compréhension de concepts avancés de programmation. Cette expérience collaborative a été un moteur de motivation et d'apprentissage continu.

J'ai remarqué une amélioration significative de mes compétences en résolution de problèmes. Chasser les bugs et optimiser le code pour les grilles de Sudoku plus complexes m'a donné une satisfaction particulière. J'ai appris à mieux anticiper les interactions entre les différentes parties du code, ce qui est crucial pour le développement de jeux logiques.