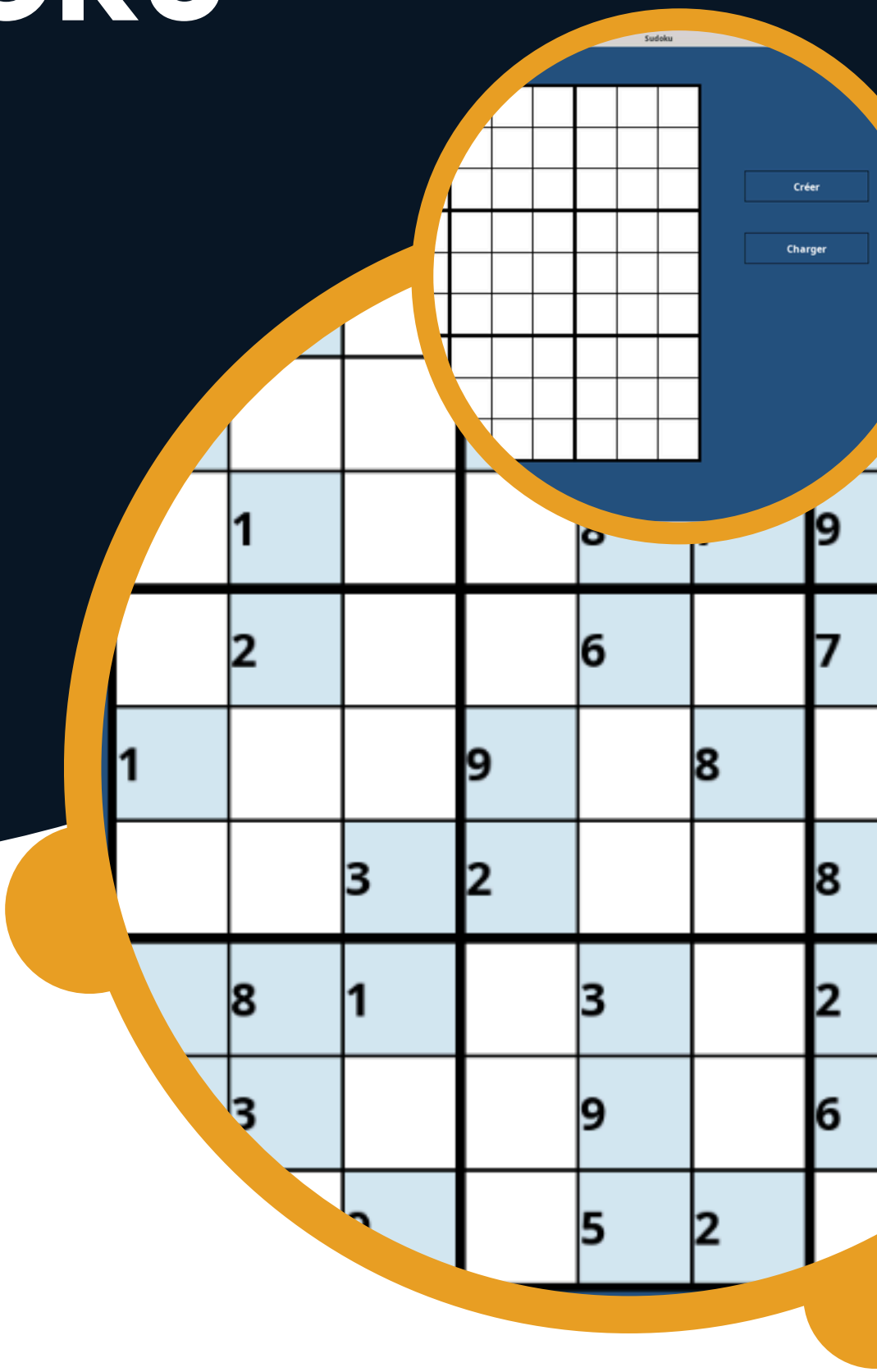
RAPPORT JAVA SUDOKU 2024



Realisé par

ABED BRIDJA

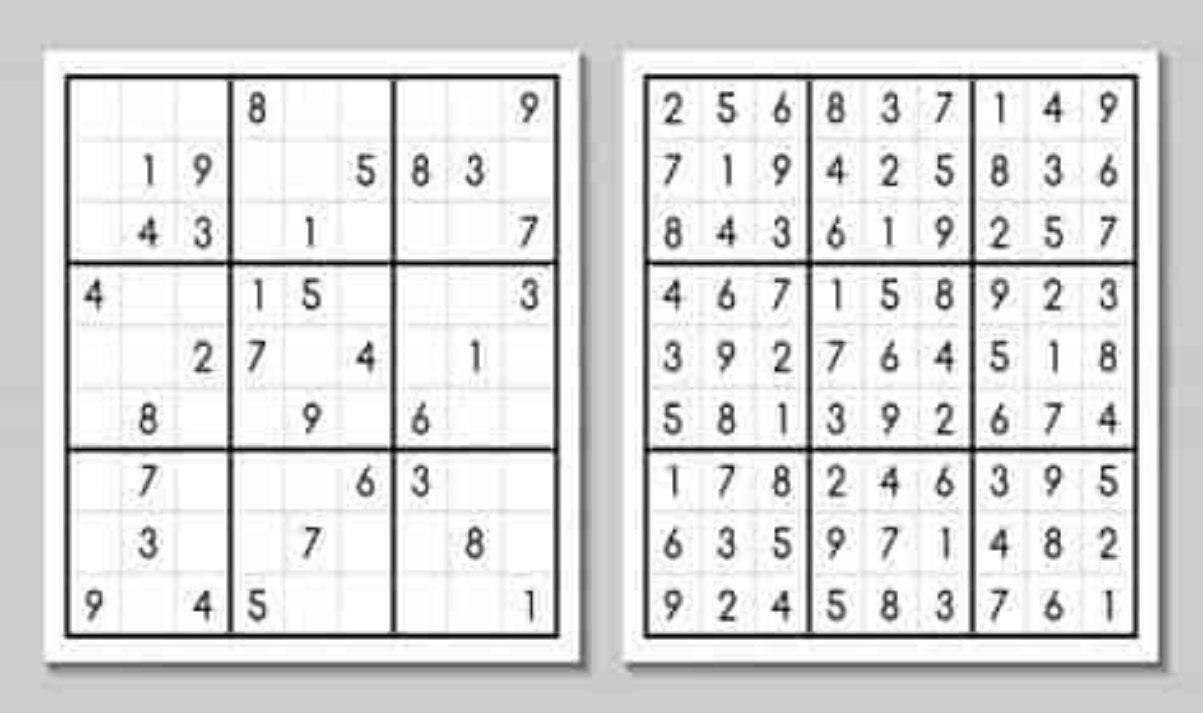
réalisé par

CHRISTOPHER DUBREUIL

SOMMAIRE

INTRODUCTION

Le Sudoku est un jeu de puzzle basé sur la logique, qui met au défi les joueurs à remplir une grille de 9x9 avec des chiffres de 1 à 9. La grille est également subdivisée en neuf régions plus petites de 3x3. L'objectif est de remplir chaque case de la grille de sorte que chaque ligne, chaque colonne et chaque région contienne tous les chiffres de 1 à 9, sans qu'il y ait de répétition d'un même chiffre dans une même ligne, colonne ou région. Les grilles de Sudoku sont généralement proposées aux joueurs avec quelques cases préremplies, et les joueurs doivent utiliser la logique pour déduire les chiffres manquants sans avoir à deviner. La création de grilles de Sudoku avec une seule solution est un processus complexe qui nécessite une attention particulière pour garantir que chaque grille présente un défi solvable par la logique plutôt que par essais et erreurs. Dans ce rapport, nous allons explorer la conception et l'implémentation d'un jeu de Sudoku en Java, en mettant l'accent sur les techniques utilisées pour générer des grilles de Sudoku et pour permettre aux joueurs de les résoudre de manière efficace.



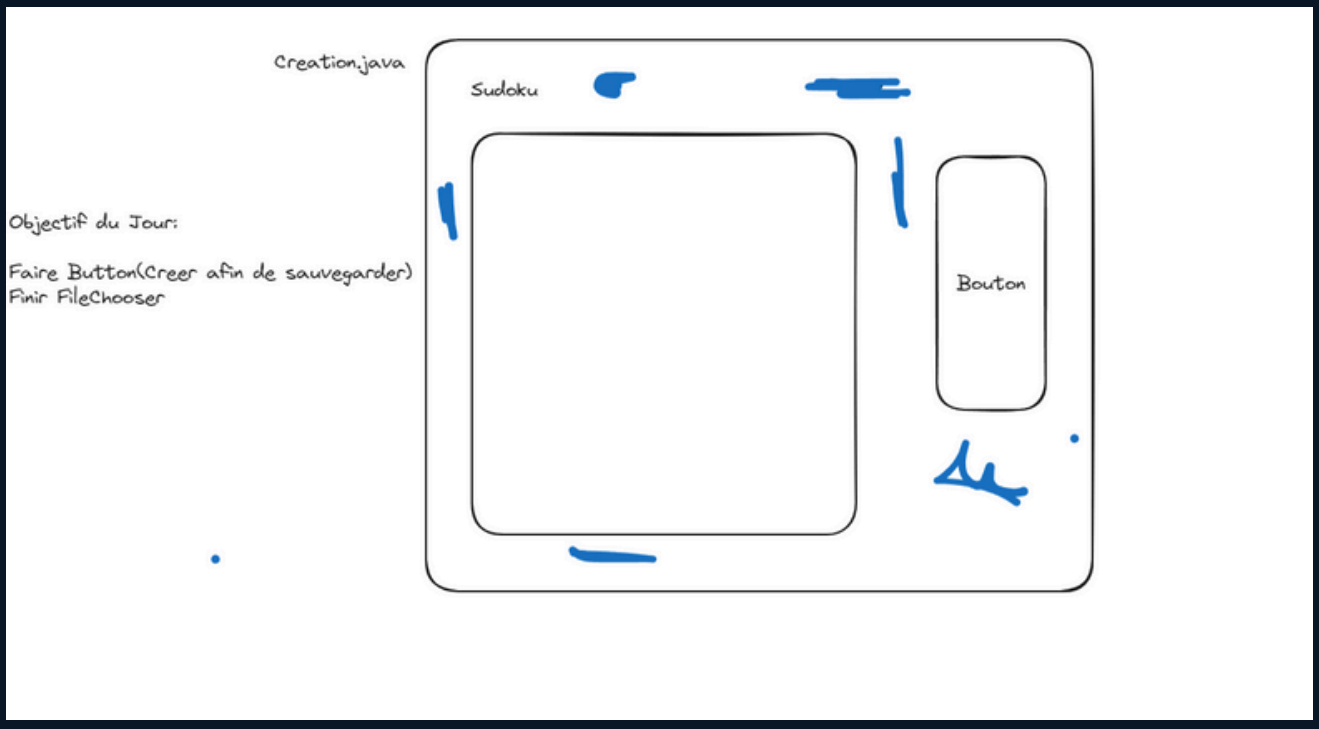
			8					9
	1	9			5	8	3	
	4	3		1				7
4			1	5				3
		2	7		4		1	
	8			9		6		
	7				6	3		
	3			7			8	
9		4	5					1

2	5	6	8	3	7	1	4	9
7	1	9	4	2	5	8	3	6
8	4	3	6	1	9	2	5	7
4	6	7	1	5	8	9	2	3
3	9	2	7	6	4	5	1	8
5	8	1	3	9	2	6	7	4
1	7	8	2	4	6	3	9	5
6	3	5	9	7	1	4	8	2
9	2	4	5	8	3	7	6	1

BRAINSTORMING

Au cours de notre période de brainstorming, nous avons mis en place un programme détaillé répertoriant toutes les tâches nécessaires à l'achèvement de notre projet. Pour assurer une gestion optimale, nous avons élaboré un emploi du temps hebdomadaire qui définit clairement les missions à accomplir pendant la semaine, réservant le week-end à la finalisation des tâches en cours et à la planification des nouvelles activités pour la semaine suivante. Nos réunions se sont déroulées à divers endroits, que ce soit dans les salles informatiques physiques ou sur la plateforme Discord, offrant ainsi une flexibilité essentielle à notre équipe. Ces sessions étaient cruciales pour discuter de l'évolution constante de notre projet, échanger des idées novatrices que nous avions conceptualisées, et revisiter les fonctionnalités que nous avions temporairement écartées, les intégrant dans un fichier commun où nous partagions des idées en mode brouillon. La collaboration ne se limitait pas à la semaine de travail régulière, car nous avons également utilisé les weekends pour des échanges approfondis sur notre progression. Ces discussions couvraient divers aspects, y compris les points de vue individuels et même les désaccords, mais notre engagement envers un objectif commun était toujours maintenu, assurant ainsi la satisfaction générale de l'équipe. Cette approche collaborative a non seulement renforcé notre cohésion, mais elle a également enrichi la qualité et la diversité des idées intégrées dans notre projet, créant ainsi une dynamique de travail productive et harmonieuse.

En voici un exemple de tache réalisé :

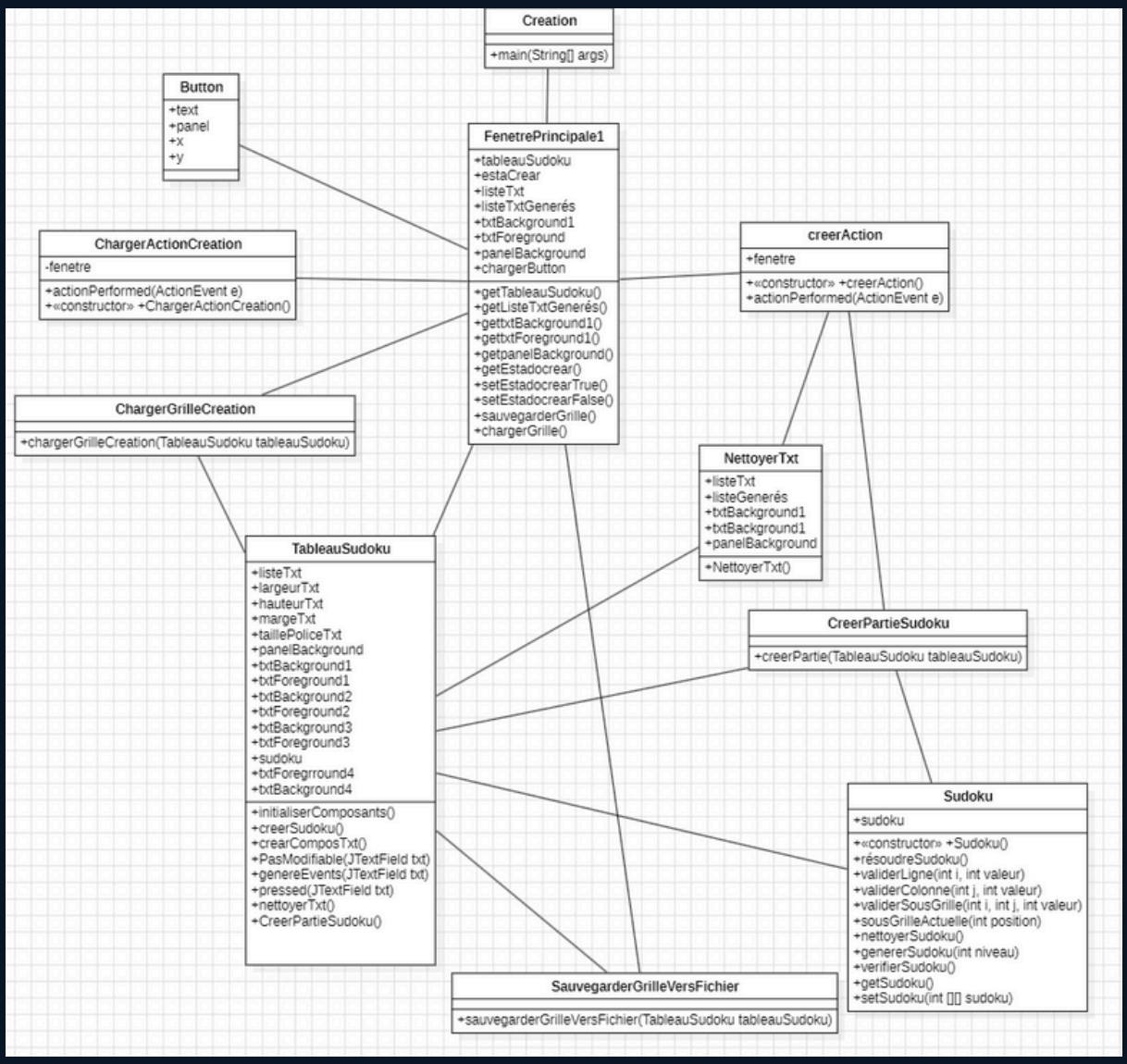


UNE PRÉSENTATION DE LA STRUCTURE DU PROGRAMME 1

Explication des Classes :

- FenetrePrincipale1 : Cette classe représente la fenêtre principale de l'application. Elle contient une grille de Sudoku (instance de TableauSudoku) et gère les interactions utilisateur telles que le chargement de grilles et la création de nouvelles grilles.
- TableauSudoku : Représente la grille de Sudoku affichée dans la fenêtre principale. Elle contient un tableau de champs de texte (JTextField) pour chaque case de la grille, ainsi que les fonctionnalités pour créer, charger et manipuler les grilles.
- Sudoku : Représente la logique métier du jeu Sudoku. Cette classe offre des fonctionnalités pour générer, résoudre et valider des grilles de Sudoku.
- CreerPartieSudoku : Cette classe fournit une fonction pour créer une nouvelle partie de Sudoku en initialisant une grille avec des chiffres valides et en la présentant à l'utilisateur.
- **ChargerActionCreation :** Cette classe implémente l'action à effectuer lors du chargement d'une grille de Sudoku. Elle est responsable de charger la grille à partir d'un fichier spécifié par l'utilisateur et de l'afficher dans la fenêtre principale.
- **Creation :** Cette classe contient la méthode **main**, qui lance l'application en créant une instance de **FenetrePrincipale1**.
- **Button :** Cette classe fournit une méthode statique **createButton** qui permet de créer des boutons avec des propriétés spécifiées telles que le texte, la police, les couleurs et la position. Elle est utilisée pour générer des boutons dans l'interface utilisateur de la fenêtre principale.
- **ChargerGrilleCreation :** Cette classe contient une méthode statique **chargerGrilleCreation** qui permet de charger une grille de Sudoku à partir d'un fichier sélectionné par l'utilisateur. Elle est responsable de lire le fichier et d'initialiser les cases de la grille en conséquence.
- **SauvegarderGrilleVersFichier :** Cette classe fournit une méthode statique **sauvegarderGrilleVersFichier** qui permet de sauvegarder la grille de Sudoku actuelle dans un fichier spécifié par l'utilisateur. Elle est responsable de l'écriture des données de la grille dans le fichier au format approprié.
- **CreerPartieSudoku :** Cette classe fournit une méthode statique **creerPartie** qui permet de créer une nouvelle partie de Sudoku à partir des données entrées par l'utilisateur dans la grille. Elle valide également les entrées de l'utilisateur et affiche des messages d'erreur le cas échéant.
- **NettoyerTxt :** Cette classe fournit une méthode **NettoyerTxt** qui permet de nettoyer les textes des cases de la grille. Elle est utilisée pour réinitialiser la grille après la fin d'une partie ou lors de la création d'une nouvelle partie.
- **creerAction :** Cette classe implémente l'interface ActionListener et fournit une méthode **actionPerformed** qui définit le comportement à exécuter lorsque l'utilisateur interagit avec les boutons de la fenêtre principale. Elle gère les actions de création d'une nouvelle partie ou de nettoyage de la grille.

UNE PRÉSENTATION DE LA STRUCTURE DU PROGRAMME 1



UNE PRÉSENTATION DE LA STRUCTURE DU PROGRAMME 2

Explication des Classes :

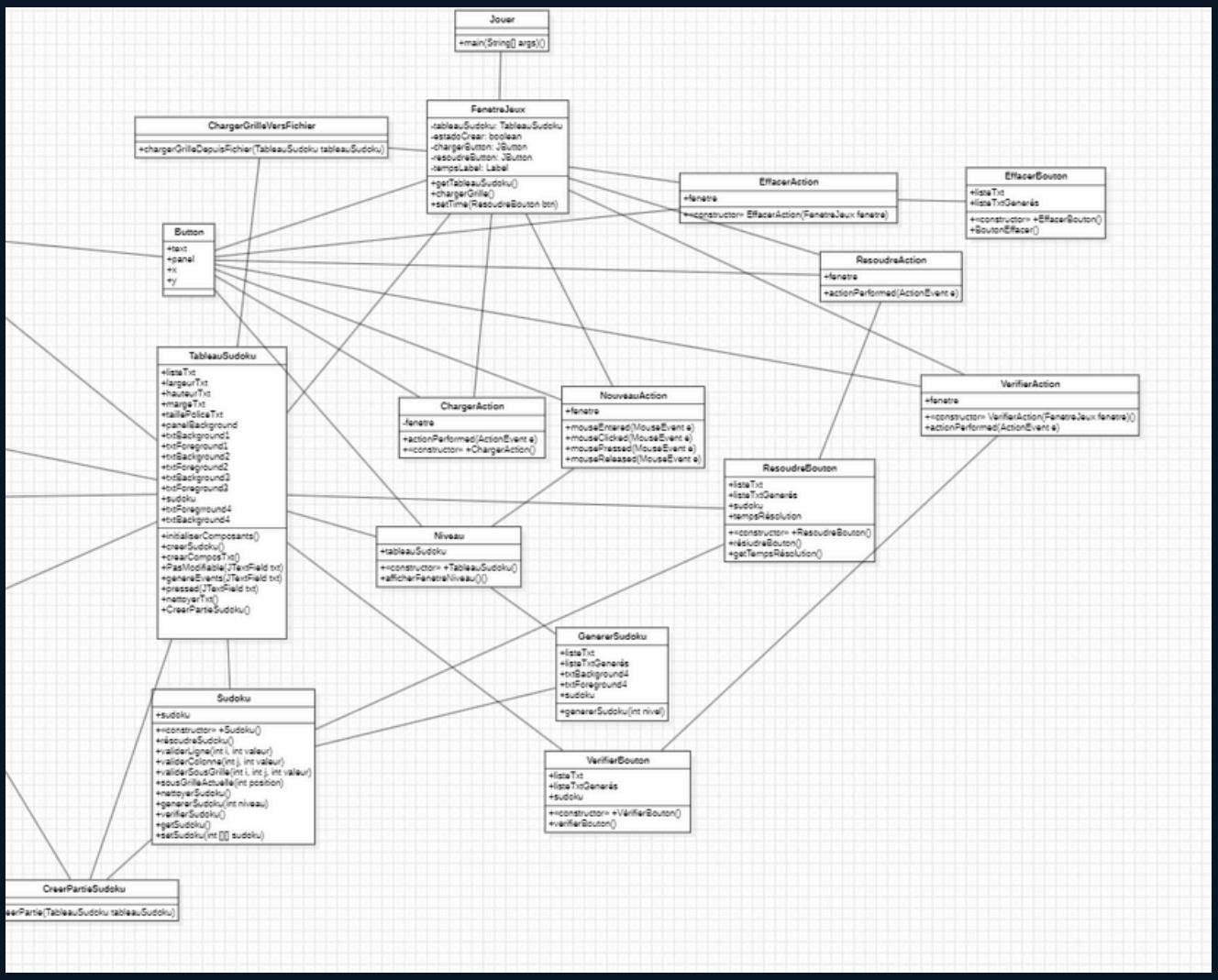
- **FenetreJeux** : Cette classe représente la fenêtre principale de l'application Sudoku. Elle contient une instance de **TableauSudoku** pour afficher la grille de jeu et gère les interactions utilisateur telles que le chargement de grilles et la résolution.
- **TableauSudoku** : Représente la grille de Sudoku dans l'interface graphique. Elle contient un tableau de champs de texte (JTextField) pour chaque case de la grille et offre des fonctionnalités pour créer, charger et manipuler les grilles.
- **Sudoku** : Gère la logique métier du jeu Sudoku. Cette classe offre des fonctionnalités pour générer, résoudre et valider des grilles de Sudoku.
- **CreerPartieSudoku** : Fournit une fonction pour créer une nouvelle partie de Sudoku en initialisant une grille avec des chiffres valides et en la présentant à l'utilisateur.
- **ChargerAction** : Implémente l'action à effectuer lors du chargement d'une grille de Sudoku. Elle est responsable de charger la grille à partir d'un fichier spécifié par l'utilisateur et de l'afficher dans la fenêtre principale.
- **ResoudreAction** : Implémente l'action à effectuer lors de la résolution d'une grille de Sudoku. Elle utilise la classe **ResoudreBouton** pour résoudre la grille et afficher le temps de résolution.
- **EffacerAction** : Implémente l'action à effectuer lors de l'effacement des valeurs des cases non générées. Elle utilise la classe **EffacerBouton** pour effectuer cette action.
- **VerifierAction** : Implémente l'action à effectuer lors de la vérification d'une grille de Sudoku. Elle utilise la classe **VerifierBouton** pour vérifier la validité de la grille.
- **NouveauAction** : Implémente l'action à effectuer lors de la création d'une nouvelle partie de Sudoku. Elle affiche une fenêtre de sélection de la difficulté et utilise la classe **GenererSudoku** pour créer une nouvelle grille.
- **GenererSudoku** : Gère la génération d'une nouvelle grille de Sudoku avec différentes difficultés. Elle utilise la classe **NettoyerTxt** pour nettoyer les textes des cases de la grille avant de générer une nouvelle grille.

UNE PRÉSENTATION DE LA STRUCTURE DU PROGRAMME 2

Explication des Classes :

- **EffacerBouton** : Fournit une méthode pour effacer les valeurs des cases non générées dans la grille de Sudoku.
- **ResoudreBouton** : Gère la résolution d'une grille de Sudoku et affiche le temps de résolution après avoir trouvé la solution.
- **VerifierBouton** : Vérifie la validité d'une grille de Sudoku et affiche un message indiquant si la grille est correcte ou non.
- **NettoyerTxt** : Fournit une méthode pour nettoyer les textes des cases de la grille, utilisée lors de la création d'une nouvelle partie ou après la fin d'une partie.
- **Button :** Cette classe fournit une méthode statique **createButton** qui permet de créer des boutons avec des propriétés spécifiées telles que le texte, la police, les couleurs et la position. Elle est utilisée pour générer des boutons dans l'interface utilisateur de la fenêtre principale.
- **Niveau** : Cette classe représente la fenêtre de sélection de la difficulté pour une nouvelle partie de Sudoku. Elle affiche une fenêtre avec des boutons pour choisir la difficulté (facile, moyen, difficile). Lorsque l'utilisateur sélectionne une difficulté, elle utilise la classe **GenererSudoku** pour créer une nouvelle grille en fonction de la difficulté choisie.
- **Jouer** : Cette classe est probablement destinée à gérer le lancement de l'application Sudoku. Elle peut contenir la méthode **main** qui initialise la fenêtre principale de l'application (**FenetreJeux**) et la rend visible.
- **ChargerGrilleVersFichier** : Cette classe fournit une méthode pour charger une grille de Sudoku à partir d'un fichier spécifié par l'utilisateur. Elle est responsable de lire les données de la grille à partir du fichier et de les charger dans la grille de Sudoku de l'interface graphique.

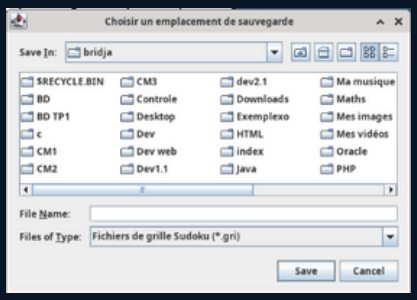
UNE PRÉSENTATION DE LA STRUCTURE DU PROGRAMME 2



FONCTIONNALITÉ

Construction de Grille à partir de zéro :

- **L'utilisateur peut démarrer avec une grille vide.**
- **Il peut remplir les cases avec des chiffres.**
- **Une fois qu'il a terminé, il peut appuyer sur le bouton "Créer" pour ouvrir une fenêtre de navigation de fichiers et enregistrer sa grille dans un fichier.**



Chargement de Grille Existante :

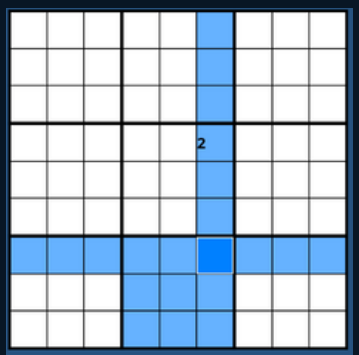
- **L'utilisateur peut également charger une grille existante en appuyant sur le bouton "Charger".**
- **L'explorateur de fichiers s'ouvre et il peut choisir un fichier au format .gri.**
- **Il peut alors modifier la grille en retirant ou en ajoutant des chiffres.**
- **Il peut sauvegarder ses modifications en appuyant sur le bouton "Créer".**

Vérification de la validité de la grille :

- **Lorsque l'utilisateur appuie sur le bouton "Créer", le programme effectue une vérification du placement des chiffres dans les lignes, les colonnes et les régions.**
- **Une fenêtre s'affiche si un chiffre est présent plusieurs fois dans la même ligne, colonne ou région.**

Mise en évidence des sélections :

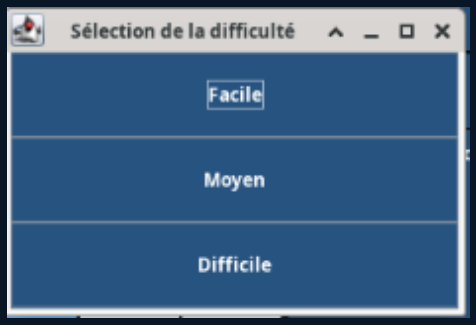
- **Lorsque l'utilisateur sélectionne une case du tableau, la ligne, la colonne et la région correspondantes deviennent bleues, facilitant ainsi la visualisation.**



FONCTIONNALITÉ

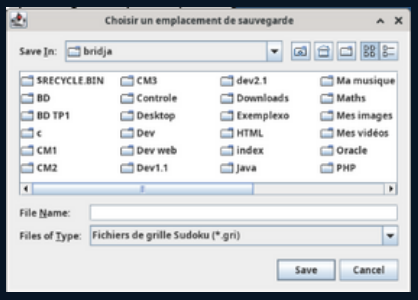
Bouton "Nouvelle Partie" :

- **Lorsque l'utilisateur survole ce bouton, trois modes de difficulté apparaissent : Facile, Moyen et Difficile.**
- **Ces modes permettent au joueur de choisir le niveau de difficulté de la grille, offrant ainsi une expérience de jeu adaptée à ses compétences et à ses préférences.**



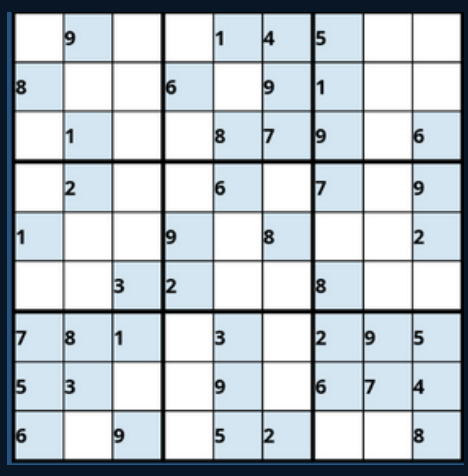
Bouton "Charger" :

- **Une fois qu'une grille a été créée dans le programme précédent, l'utilisateur peut l'importer dans le jeu en appuyant sur ce bouton.**
- **Cela lui permet de jouer à la grille qu'il a créée ou modifiée dans le programme de construction de grilles.**



Affichage des Chiffres Bloqués :

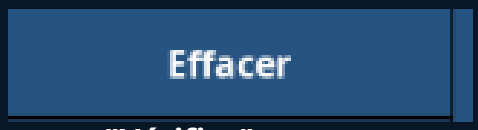
- **Lorsqu'une grille est importée ou lorsque le joueur choisit l'un des trois modes de difficulté, certains chiffres apparaissent dans la grille et ne peuvent pas être effacés.**
- **Ces chiffres bloqués sont reconnaissables grâce à un fond bleu ciel présent dans la case du Sudoku.**
- **Cette couleur de fond spécifique permet au joueur de distinguer facilement les chiffres qui sont fixés et ne peuvent pas être modifiés.**
- **Ces chiffres bloqués sont là pour garantir que le joueur ne peut pas tricher en modifiant les chiffres de départ.**



FONCTIONNALITÉ

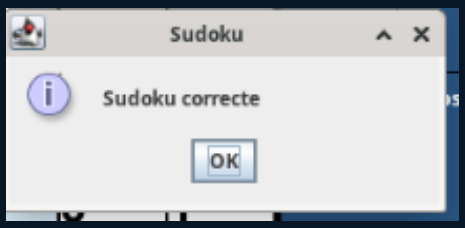
Bouton "Effacer" :

- **Ce bouton permet d'effacer uniquement les chiffres que le joueur a placés dans les cases.**
- **Les chiffres présents dans les modes de difficulté et ceux importés via le bouton "Charger" ne sont pas affectés.**



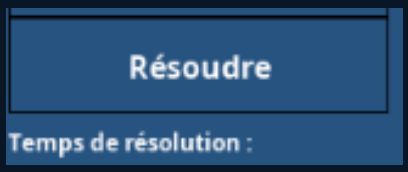
Bouton "Vérifier" :

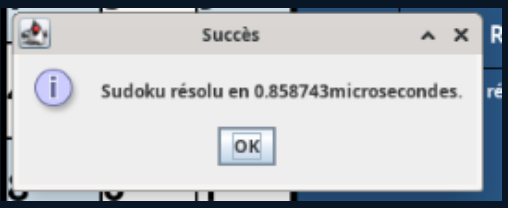
- **Ce bouton permet au joueur de vérifier s'il a terminé la grille correctement.**
- **Le programme vérifie si tous les chiffres sont présents dans chaque case et s'ils respectent l'unicité des chiffres dans les lignes, colonnes et régions.**



Bouton "Résoudre" :

- **Lorsque le joueur importe une grille ou choisit l'un des trois modes de difficulté, il peut utiliser ce bouton pour résoudre automatiquement la grille.**
- **Une fenêtre s'affiche, indiquant le temps de résolution de la grille en microsecondes.**





Possibilité de Réflexion :

- **Pour aider à la réflexion du joueur, il est possible de placer jusqu'à 4 chiffres différents dans la même case.**
- **Cela permet au joueur d'explorer différentes possibilités et stratégies sans effacer les chiffres déjà placés.**

L'ALGORITHME QUI RÉSOUD LES GRILLES

L'algorithme utilisé dans la méthode "résoudreSudoku()" dans le fichier Sudoku.java pour résoudre les grilles de Sudoku est une implémentation de la technique de recherche récursive avec backtracking*.

Voici une explication détaillée de son fonctionnement :

1. **Parcours de la grille :** L'algorithme commence par parcourir la grille de Sudoku, case par case, en utilisant deux boucles imbriquées pour itérer à travers chaque ligne et chaque colonne.
2. **Recherche d'une case vide :** Lorsqu'une case vide est rencontrée (représentée par la valeur 0 dans la grille), l'algorithme considère cette case comme une candidate pour placer un chiffre.
3. **Attribution de valeurs :** L'algorithme tente d'attribuer une valeur possible (de 1 à 9) à la case vide en utilisant une boucle itérative. Pour chaque valeur possible, il vérifie si cette valeur respecte les contraintes du Sudoku : aucune répétition du même chiffre dans la même ligne, la même colonne ou la même sous-grille 3x3.
4. **Validation des contraintes :** Avant d'attribuer une valeur à une case vide, l'algorithme vérifie trois contraintes :
   - La valeur n'est pas déjà présente dans la même ligne.
   - La valeur n'est pas déjà présente dans la même colonne.
   - La valeur n'est pas déjà présente dans la même sous-grille 3x3.
5. **Récursion :** Si une valeur peut être attribuée à la case vide sans violer les contraintes, elle est affectée à la case, et la méthode est appelée récursivement pour traiter la prochaine case vide. Cette récursion se poursuit jusqu'à ce que toutes les cases de la grille soient remplies avec succès ou qu'aucune valeur possible ne puisse être attribuée à une case vide.
6. **Backtracking :** Si aucune valeur possible ne peut être attribuée à une case vide sans violer les contraintes, l'algorithme revient en arrière (backtracking) en réinitialisant la valeur de la case actuelle à 0 et en essayant une autre valeur possible. Cela permet à l'algorithme d'explorer différentes possibilités et de revenir en arrière si une erreur est détectée.
7. **Terminaison :** L'algorithme continue de parcourir la grille et de traiter chaque case vide de manière récursive jusqu'à ce que toutes les cases soient remplies avec succès (et que la grille soit résolue) .

*backtracking = Le backtracking est une méthode algorithmique qui explore séquentiellement les solutions possibles d'un problème, revenant en arrière lorsque nécessaire pour corriger les erreurs.

CONCLUSION PERSONNELLE

Abed Bridja

La réalisation du projet "Sudoku" a été une expérience enrichissante, marquant une étape significative dans mon parcours de programmation. Mon ressenti à l'égard de ce projet se situe entre l'enthousiasme face à sa concrétisation et la reconnaissance des défis qui ont été difficiles à surmonter. Au cours de ce projet, j'ai été confronté à diverses difficultés, qu'il s'agisse de la gestion des structures de données ou de la logique de résolution des grilles. Ces défis ont été autant de tremplins pour renforcer ma compréhension des concepts de programmation, me confrontant à des problématiques concrètes et stimulantes. Néanmoins, chaque obstacle surmonté a été une victoire personnelle. La résolution de problèmes m'a poussé à explorer de nouvelles approches, à affiner mes compétences, et à approfondir ma compréhension du langage Java, constituant ainsi un moyen concret d'appliquer les connaissances acquises. Ces moments d'incertitude ont été essentiels pour ma croissance en tant que développeur en herbe, m'invitant à repousser mes limites et à rechercher des solutions créatives. En fin de compte, le projet "Sudoku" a été bien plus qu'une simple réalisation technique. Il a été une aventure, une exploration personnelle de mes compétences et de ma créativité.

Christopher Dubreuil

La réalisation du Sudoku a été, selon moi, très différent du snake programmé en C. J'ai pu découvrir, cette fois-ci, le travail d'équipe comparé à mon ancien binôme. J'appréhendais tout de même la réalisation de ce projet puisque le langage Java est, selon moi, plus compliqué. Ainsi, rentrer dans la conception du projet s'est avéré difficile et j'ai bénéficié de l'aide de mon binôme. Grâce à tout ça, nous avons pu programmer la quasi-totalité de nos idées et des fonctionnalités demandées.