Matthieu Carriço
Paul Minguet

Groupe 3

Projet Tutoré APL 2.1 2019-2020

- Sudoku -

Sommaire

Introduction :

Pour ce projet, nous avons eu pour consigne de coder un jeu de Sudoku. Ce jeu a une origine très ancienne. Il est inspiré du « carré latin », lui-même inspiré du « carré magique » chinois. Le sudoku que nous avions à faire a été, lui, créé en 1979 par Howard Garns et est apparu dans les magazines *Dell Pencil Puzzles* et *World Games* sous le nom de *Number Place*. Le principe du jeu est simple : nous disposons d'une grille de 9x9 cases plus ou moins complétée de chiffres compris entre 1 et 9. Le but est de remplir toute cette grille en faisant attention aux contraintes suivantes :

- Il ne doit pas y avoir deux fois le même chiffre sur la même ligne
- Il ne doit pas y avoir deux fois le même chiffre sur la même colonne
- Il ne doit pas y avoir deux fois le même chiffre dans le même « bloc »

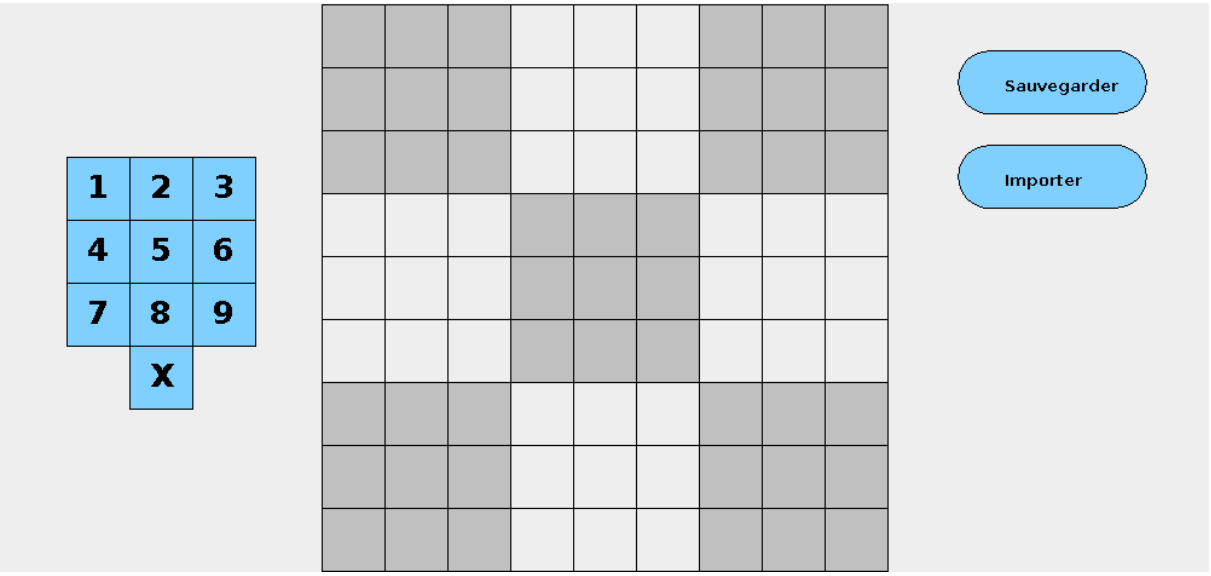
Nous devons aussi pouvoir mettre en indice et exposant, à droite et à gauche de chaque case vide, des chiffres sur lesquels nous hésitons.

Notre programme :

Lorsque le joueur lance le programme, il trouve un menu dans lequel il a le choix entre créer une grille et résoudre une grille. Le bouton *nouveau* permet à l'utilisateur de créer une grille afin de la résoudre après ou bien afin de la faire résoudre plus tard par un algorithme. Le bouton *ouvrir* permet à l'utilisateur de jouer au sudoku, donc de résoudre une grille. (cf. capture n°1)

Capture d'écran n°1 : menu

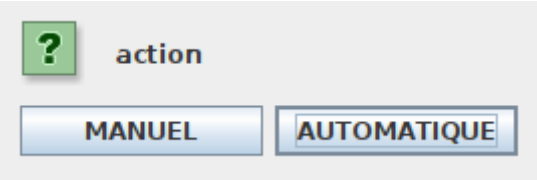
Lorsque l'utilisateur clique sur le bouton *nouveau*, une fenêtre s'ouvre avec une grille vide dans laquelle il peut placer des valeurs en cliquant sur la case qu'il souhaite sélectionner (cf. capture n°2). Il doit cependant respecter les contraintes du Sudoku (il ne doit pas y avoir deux fois la même valeur sur la même ligne, sur la même colonne ou sur le même bloc). Une fois la grille créée, il peut la sauvegarder en appuyant sur le bouton *Sauvegarde*. Celui-ci ouvre une fenêtre de dialogue et permet de sauvegarder la grille dans le répertoire voulu par l'utilisateur (cf. capture n°3). Celui-ci peut aussi importer une grille afin de la modifier en cliquant sur le bouton *Importer* : une autre fenêtre de dialogue s'ouvre et lui permet de sélectionner l'emplacement de la grille à ouvrir. Lorsque l'utilisateur clique sur la grille à ouvrir, celle-ci apparaît sur la fenêtre et il peut alors l'éditer.



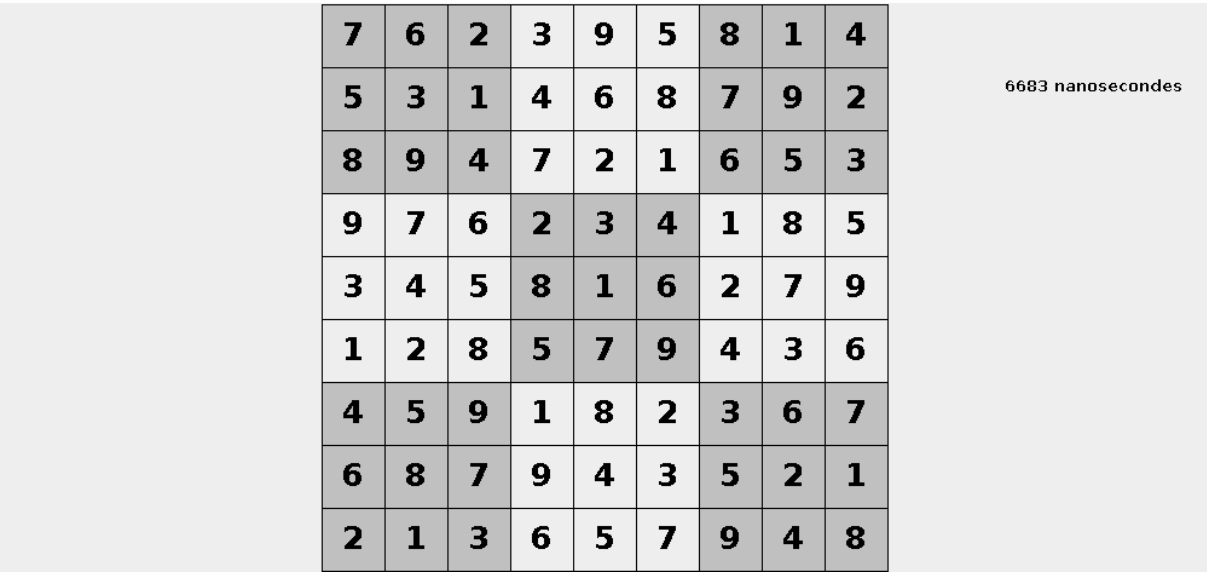
Capture d'écran n°2 : grille création

Capture d'écran n°3 : sauvegarde

Si le joueur choisit de cliquer sur le bouton *ouvrir* dans le menu, il peut choisir entre résoudre manuellement ou automatiquement une grille (cf. capture n°4). Lorsqu'il clique sur un des deux boutons, une fenêtre de dialogue lui demande de sélectionner le fichier à ouvrir (la grille qu'il veut résoudre). S'il veut la résoudre automatiquement, une fenêtre s'ouvre avec la grille choisie, résolue et le temps qu'il a fallu à l'ordinateur pour la résoudre (cf. capture n°5).

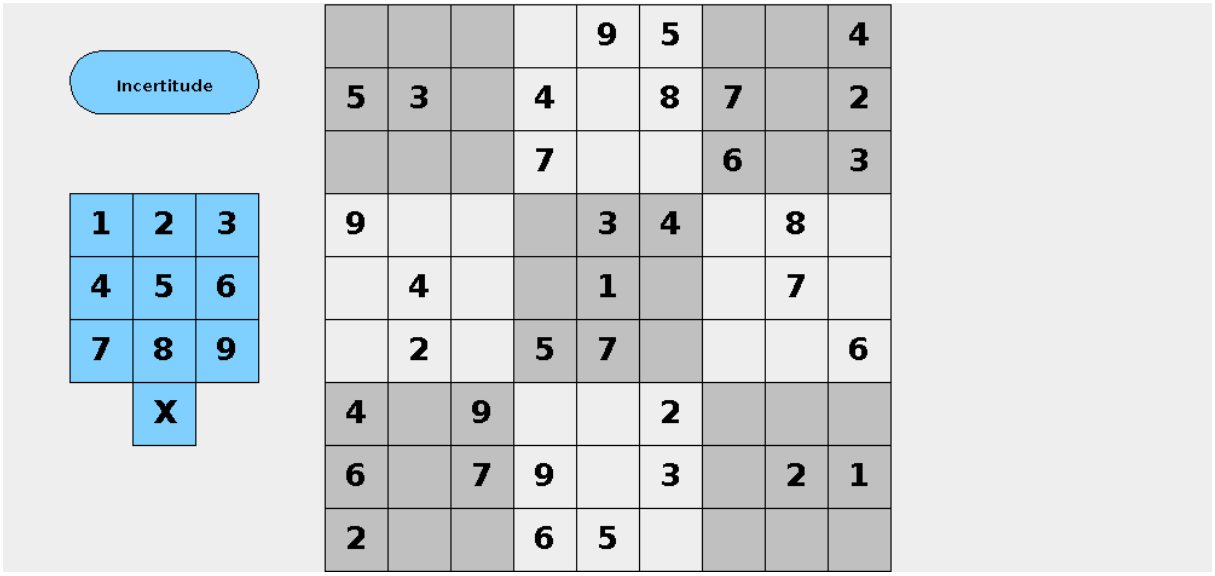


Capture d'écran n°4 : menu automatique/manuel

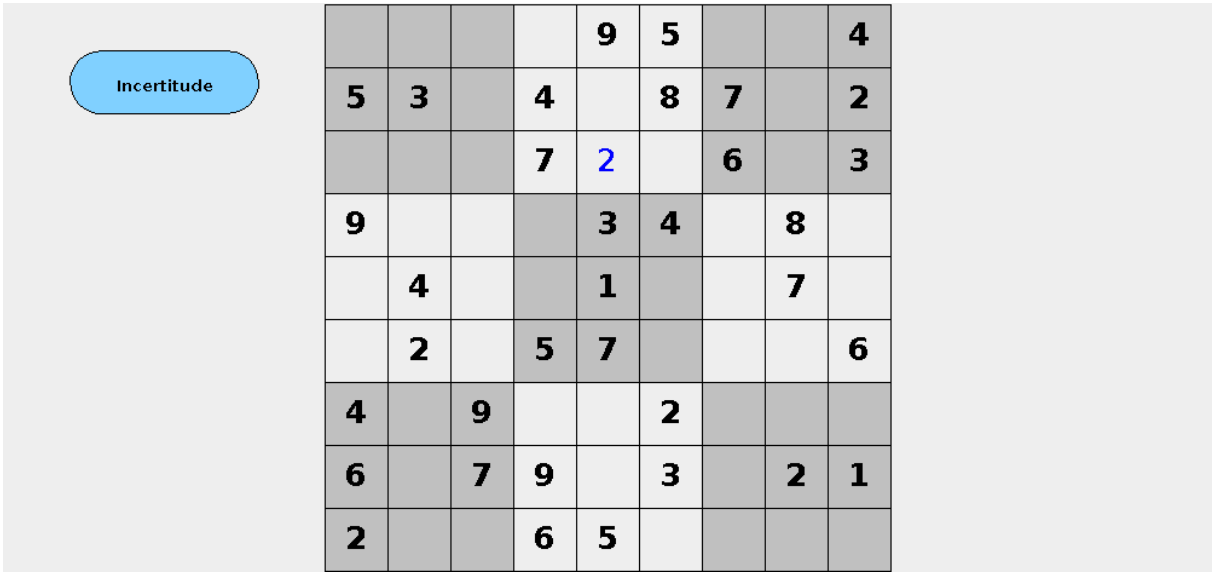


Capture d'écran n°5 : grille résolue automatiquement

Si le joueur souhaite résoudre manuellement la grille, lorsqu'il ouvre un fichier, celle-ci apparaît sur la fenêtre. Une fois dans le jeu, le joueur peut cliquer sur une case afin d'afficher un tableau de valeurs (de 1 à 9) à gauche de la grille afin de sélectionner la valeur à mettre dans la case (cf. capture n°6). Une fois que l'utilisateur clique sur une valeur dans le tableau, celle-ci est réécrite dans la grille à l'emplacement où il a cliqué. Cette valeur est de couleur bleue afin que celui-ci puisse bien différencier les valeurs de base des siennes. (cf. capture n°7)

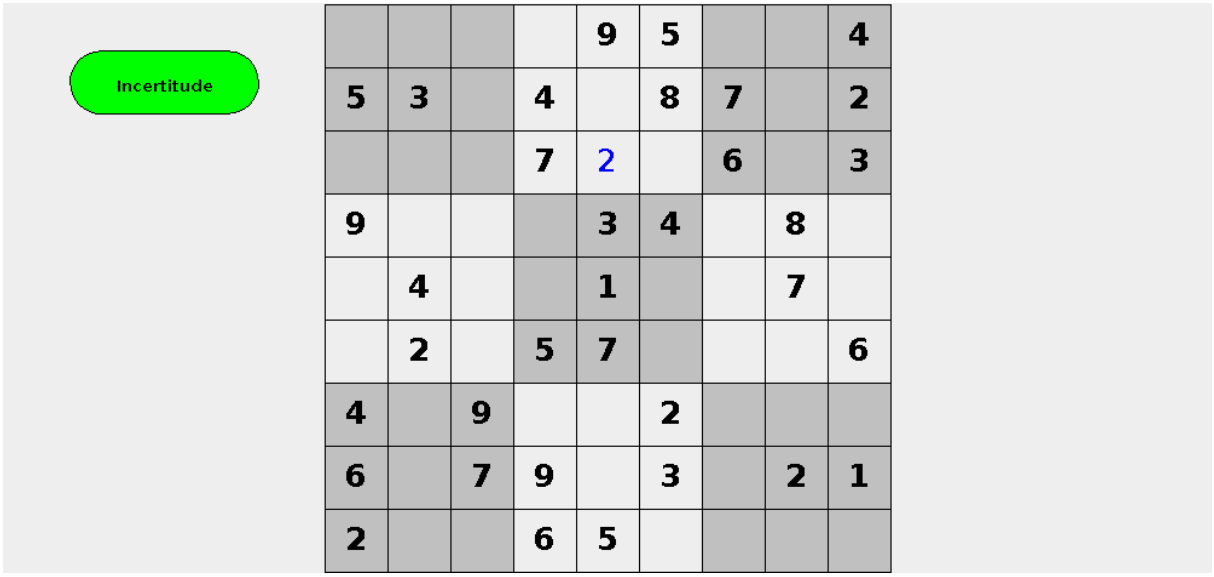


Capture d'écran n°6 : tableau de valeurs

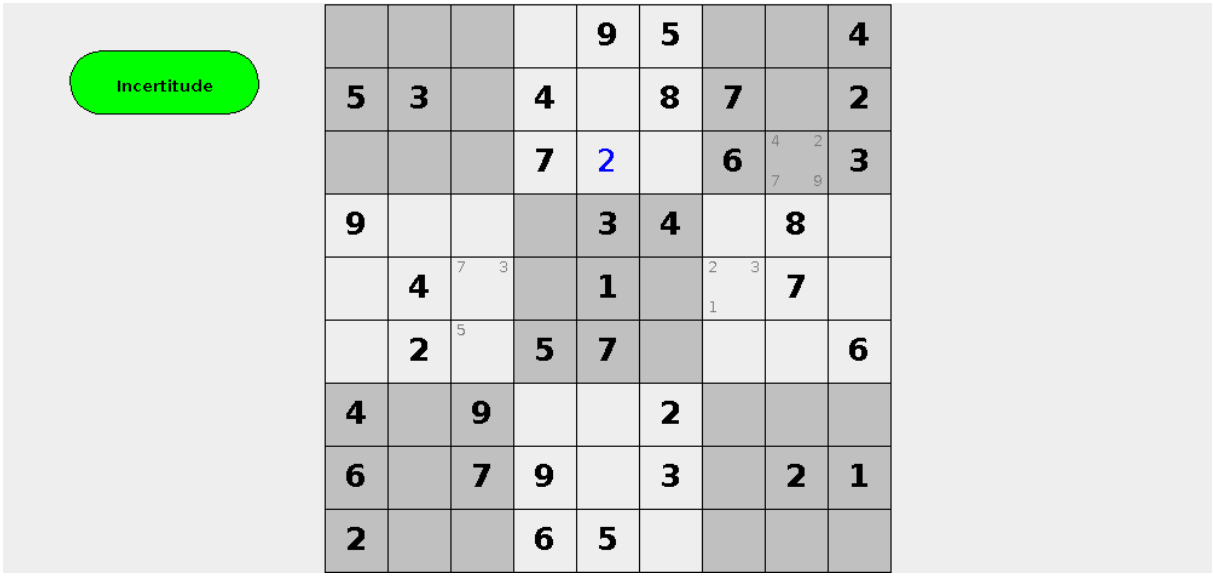


Capture d'écran n°7 : valeur ajoutée dans la grille

Si le joueur n'est pas sûr de la valeur d'une case, il peut cliquer sur le bouton *Incertitude*. Une fois cliqué, celui-ci devient vert (cf. capture 8) et lorsque le joueur clique sur une valeur dans le tableau, celle-ci se place dans la grille comme valeur « incertaine » dans l'ordre : exposant gauche puis droit puis indice gauche puis droit. (cf. capture 9)

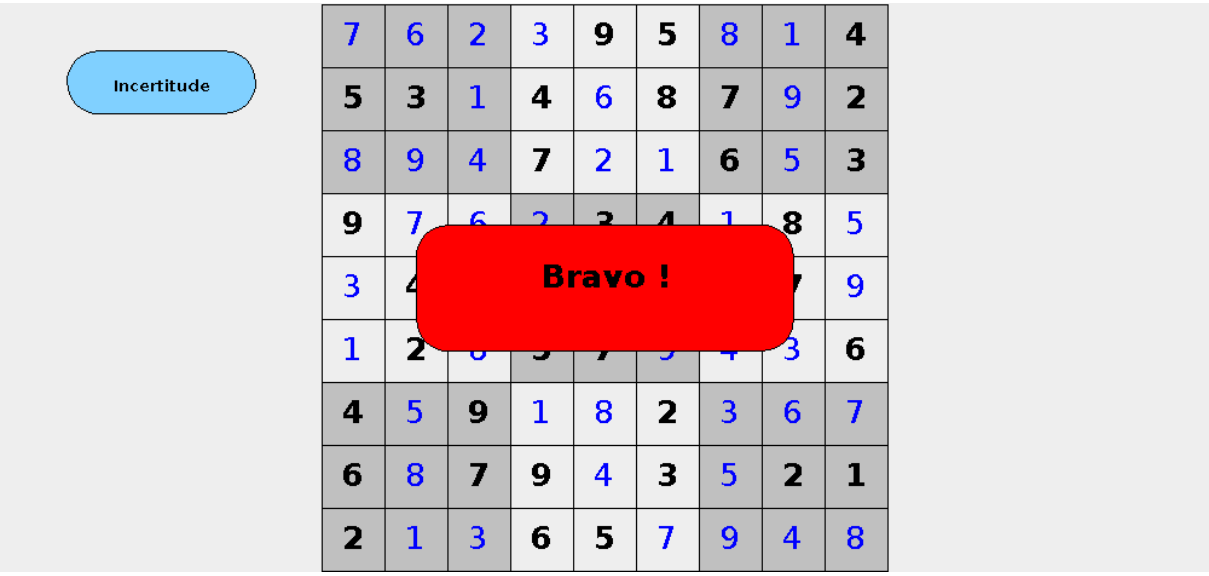


Capture d'écran n°8 : bouton incertitude cliqué



Capture d'écran n°9 : affichage exposants

Dès que l'utilisateur a fini une grille en mode manuel, alors un message s'affiche au milieu de l'écran avec « Bravo ! » pour féliciter le joueur (cf. capture n°10).



Capture d'écran n°10 : fin d'une grille

Présentation de la structure du programme :

Notre programme est composé de 6 classes :

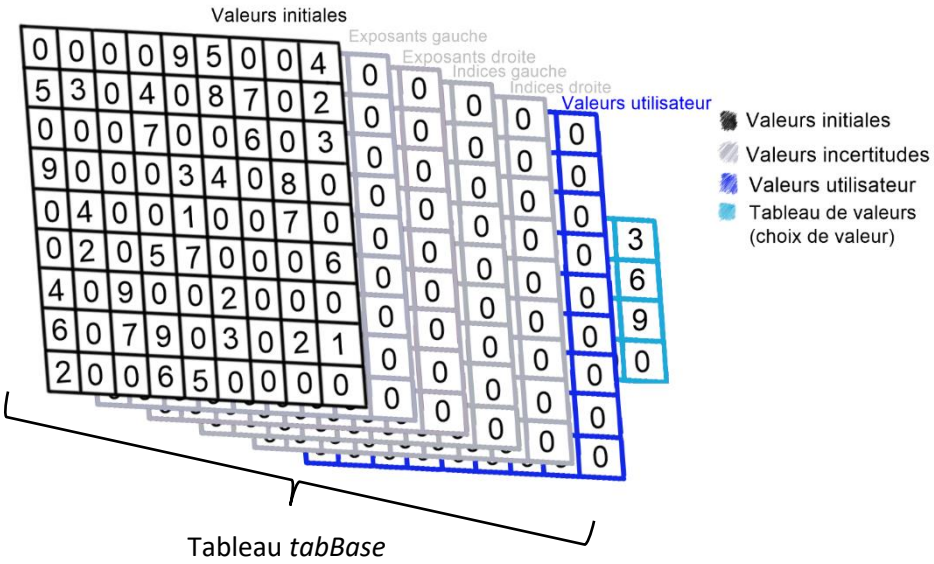
- `Main`
- `Creer` qui servira à la création manuelle de grilles
- `Grille` qui servira à la résolution manuelle de grilles
- `Grilleauto` qui servira à la résolution automatique de grilles
- `Solveur` qui contient l'algorithme de résolution des grilles et les vérifications des contraintes
- `Tableau` qui sert de stockage aux différents tableaux utilisés pour l'affichage de la grille et aux sauvegardes et importations.

Notre classe `Main` est notre classe centrale c'est de là que nous exécutons nos deux programmes. Dans cette classe nous trouvons le nécessaire pour l'affichage des menus et donc la redirection vers la classe correspondant à ce que l'utilisateur veut effectuer.

`Solveur` est une classe qui contient les méthodes permettant la résolution avec l'algorithme de résolution. Nous pourrons ainsi y trouver les 3 méthodes de vérification pour savoir si un chiffre se trouve respectivement sur une ligne, une colonne ou sur un bloc du Sudoku.
On pourra aussi trouver dans cette classe la méthode de résolution du Sudoku ainsi que des méthodes *get* et *set* afin de pouvoir permettre à d'autres classes de modifier la grille de Sudoku.

`Grilleauto` est la classe qui sert simplement à afficher une grille et un *timer*. Dans notre programme on l'utilise exclusivement pour afficher la grille résolue ainsi que le temps mis à la parcourir, temps que nous avons décidé de laisser en nanosecondes étant donné le peu de temps que l'on met à résoudre les sudokus.

Le fichier `Tableau.java` nous sert, comme précisé ci-dessus, à stocker les valeurs de la grille. Nous avons fait un tableau à trois dimensions dans lequel la première servira à préciser de quelles « valeurs » nous parlons (donc si nous parlons de celles initialement dans la grille, celles qu'a rentré l'utilisateur, celles que l'utilisateur a rentré en tant que valeurs « incertaines » ou bien celles du tableau de choix de la valeur à rentrer- cf schéma explication ci-après), la deuxième pour préciser la ligne et la troisième, la colonne.



Ce schéma représente le tableau *tabBase* après le chargement de la grille `exemple.gri`.

Ensuite, nous avons un tableau à deux dimensions qui nous permet de représenter la grille avec les valeurs initiales et celles de l'utilisateur dans le même tableau, notamment pour les vérifications des contraintes lors du choix de la valeur.

Enfin, dans ce même fichier, nous avons écrit les méthodes pour récupérer des valeurs dans les tableaux *tabBase* et grille, ainsi que celles pour sauvegarder et importer une grille.

Dans le fichier `Grille.java`, on a utilisé beaucoup de graphique ainsi que des calculs de coordonnées : tout d'abord, on va dessiner une case de taille déterminée par la taille de la fenêtre. Ensuite, on va récupérer la première valeur du tableau *tabBase* (pour les valeurs initiales) et on va l'afficher sous forme de String au milieu de la case dessinée au-dessus. Si la valeur est un « 0 », on ne l'affiche pas, sinon on l'affiche. Ensuite, on dessine la valeur qui se trouve au même emplacement dans les quatre « tableaux » des valeurs des incertitudes et celle dans le « tableau » de l'utilisateur. Lors du premier chargement, ces deux dernières étapes n'afficheront rien puisque l'utilisateur n'aura pas encore rentré de valeur, mais à partir du moment où le joueur rentrera une valeur, il faudra la dessiner pour qu'il puisse la voir. On répète ce processus jusqu'à avoir chargé tout le tableau et dessiné toute la grille.

Une fois que la grille est dessinée, on dessine le bouton I*ncertitude*.

Après cela, nous allons dessiner, mais nous ne l'afficherons pas encore, le tableau de sélection de valeur sur la gauche. Lorsque l'utilisateur cliquera dans la grille, la variable *addEtat* se mettra à 1 et nous afficherons donc ce tableau.

Nous gérons ensuite les interactions avec la souris :

Lorsqu'elle est cliquée, nous récupérons les coordonnées x et y de la souris au moment du clic (nous enlevons 30 pixels à la coordonnée y de la souris car elle prend en compte le bandeau de la fenêtre).

Ensuite, à l'aide d'une condition et de calcul, nous déterminons si le clic se situe sur la grille. Si oui, on affiche le tableau à gauche, sinon on détermine si le clic se situe sur le bouton *Incertitude*. Si le clic se situait dans la grille, alors on affiche le tableau à gauche et on attend que l'utilisateur choisisse la valeur qu'il veut rentrer. Une fois qu'il a choisi sa valeur, on détermine à l'aide de calculs quelle valeur l'utilisateur a choisi puis on rentre cette valeur dans le « tableau » utilisateur et on l'affiche à l'écran. Si l'utilisateur avait cliqué auparavant sur le bouton *Incertitude*, alors on rentre cette valeur dans un des « tableaux » incertitude en fonction du remplissage des précédents, puis on les affiche.

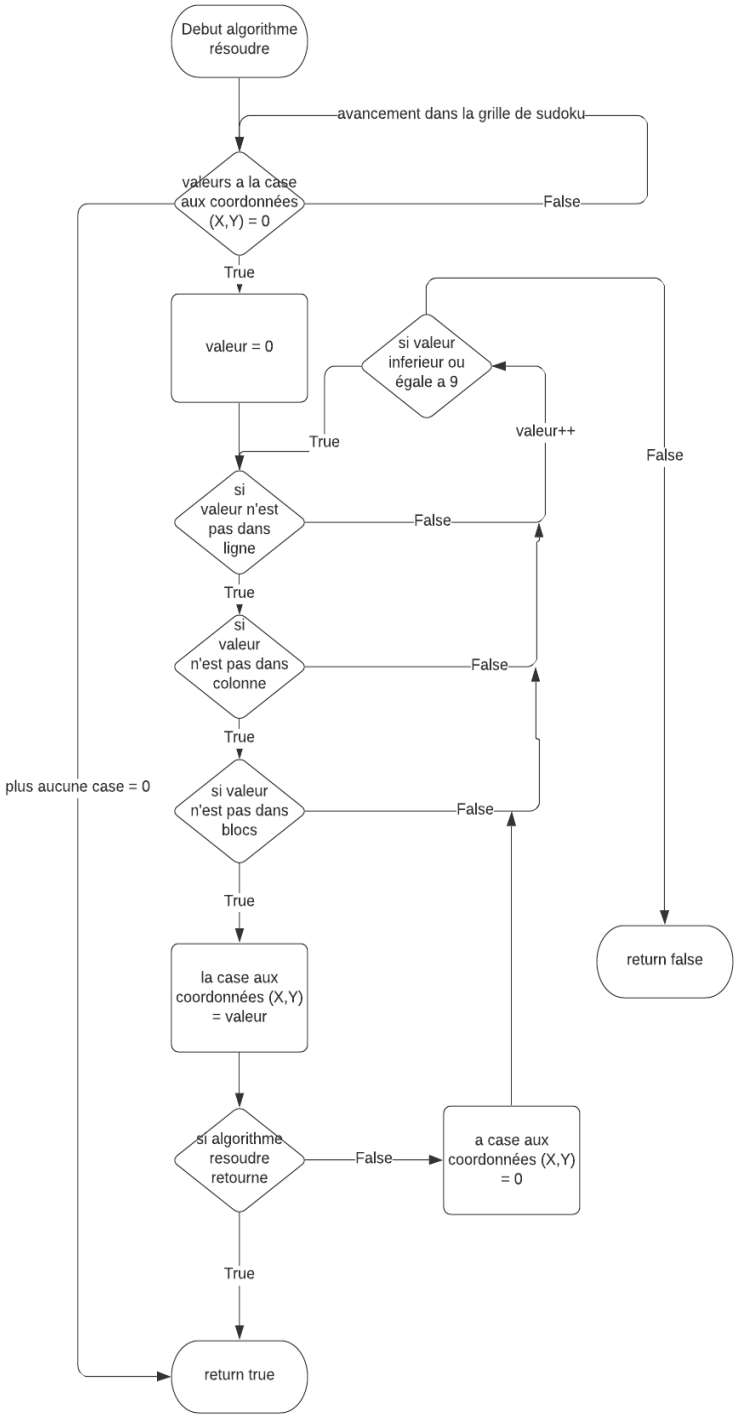
Notre fichier `Creer.java` fonctionne sur le même principe que `Grille.java`, mais nous avons ajouté des boutons *Sauvegarde* et *Importation*, et nous avons supprimé le bouton *Incertitude* car il ne sert à rien lors de la création d'une grille. De plus, lorsque l'utilisateur choisit une valeur, on écrit dans le « tableau » initial et non utilisateur. Ainsi, lorsque l'utilisateur clique sur le bouton *Sauvegarde*, alors on récupère toutes les valeurs du « tableau » initial et on les convertit en hexadécimale avant de les entrer dans un fichier choisi par l'utilisateur. Lorsque celui-ci décide d'importer une grille, on fait le processus inverse, on lit les valeurs hexadécimales, on les transforme en int en les lisant puis en string pour les entrer dans le « tableau » initial et dans la grille.

Présentation de l'algorithme de résolution de sudoku :

L'algorithme consiste en la méthode du Backtracking : nous parcourons notre grille et si nous tombons sur une case valant 0, alors nous testons pour chaque valeur si elle est présente sur la ligne, la colonne ou bien le bloc. Si ce n'est pas le cas alors nous associons la première valeur trouvée (maximum 9) à la case. Si la valeur de la case ne peut pas être comprise entre 1 et 9 alors la méthode retourne *False*.

Nous relançons notre méthode dans une condition qui vérifie ce que retourne notre méthode. Si celle-ci retourne *false* alors on met à 0 la case de la grille de sudoku et on reprend les tests de valeurs (avec toujours 9 comme maximum !).
Si la méthode retourne *True* alors on retourne *True* dans la condition.
Si on ne trouve aucune case dans la grille égale à 0 alors la méthode retourne *True*.

Afin de mieux comprendre l'algorithme vous pouvez consulter l'algorigramme placé à droite.



Conclusions personnelles :

Matthieu :

La réalisation de ce projet m'a permis de réaliser deux grandes choses : la première est l'importance de la communication dans un projet (surtout en cas de télétravail) et la deuxième chose est la difficulté que ça peut être d'expliquer un algorithme, difficultés qui se décuplent lorsque celui-ci est récursif et ce malgré la simplicité de celui que nous avons utilisé. Sinon c'était mon deuxième projet avec Paul Minguet, ce qui nous a permis de continuer de faire ce qu'on avait déjà fait lors du premier projet en APL, soit donner la partie du travail à celui qui maitrise le moins bien cette dite partie. Nous mettons certes plus de temps à le faire mais on sort des projets avec l'acquisition de compétences que nous n'avions pas forcément initialement.

Paul :

Pour ce deuxième projet que j'ai fait avec Matthieu Carriço, j'ai eu peur que le temps donné pour le réaliser soit un peu court. Le jeu en lui-même n'a pas été très long à faire, ce qui nous a pris le plus de temps c'était la sauvegarde et l'importation en hexadécimale, ainsi que l'algorithme. Même si cela ne nous a pas forcément pris très longtemps à faire le jeu, personnellement, cela m'a quand même permis de comprendre quelques détails que je n'avais pas compris dans le java. Ensuite, ça nous a permis, avec Matthieu, de voir ce qui pouvait être compliqué dans les projets à deux, comme la mise en commun de nos programmes, la coordination, les horaires pour se retrouver, etc… Ce projet nous a donc apporté des compétences et du savoir sur le travail de groupe en informatique.