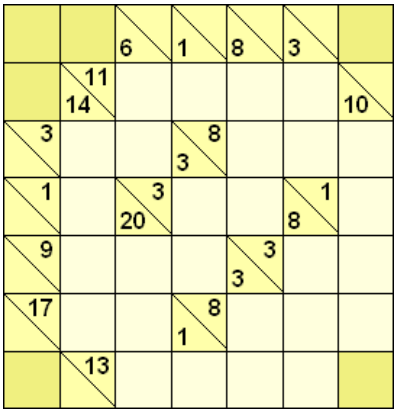
Université de Strasbourg

Projet POO - JAVA

Cadre de probléme :

Programmation d'un jeux en java : KAKURO



Fait par :

- FERGUI Samy

Encadré par :

- Frey Gabriel

Année universitaire : 2021/2022

1. Introduction :

Le Kakuro est un jeu de type « nombres fléchés ». Il s'agit d'une grille comportant des cases remplies non modifiables (cases noires), des cases contenant des nombres, des cases vides à remplir avec un nombre et des cases d'indication pouvant contenir deux valeurs, la somme des cases contiguës de la colonne sous la case d'indication et/ou la somme des cases contiguës de la ligne à droite de la case d'indication.

Pour compléter une grille, il s'agit de remplir toutes les cases vides avec des chiffres, de 1 à 9, de telle sorte à ce que les sommes indiquées dans les cases d'indication soient respectées.

Une contrainte supplémentaire est que colonnes ou lignes contiguës ne peuvent contenir qu'une seule fois chaque nombre de 1 à 9.

---

2. Conception et réalisation

L'idée de jeux à réaliser est de pouvoir générer une grille KAKURO d'une façon totalement aléatoire, donner la possibilité au joueur de définir le niveau de jeux (facile, moyen, difficile), pouvoir remplir la grille sous quelques restrictions définis (possibilité de taper un seul caractère numérique), donner un Hint pour le joueur, checker la grille rempli par ce dernier et enfin proposer une solution pour la personne qui joue (solver).

A. Solver :

Pour rendre l'implémentation plus facile, je pars de l'idée qu'on dispose d'une grille KAKURO qui a au moins une solution . Pour ce, je génère une grille KAKURO aléatoirement tout en ayant une solution pour cette dernière, que je sauvegarde bien sûr quelque part. Le nombre de cases remplies avec des nombres est généré d'une manière aléatoire selon le niveau choisi par le joueur, l'emplacement de ces dernières est aussi pris en compte par le hasard !
Pour afficher la solution de la grille donnée à l'utilisateur, il suffit de cliquer sur le bouton solver.

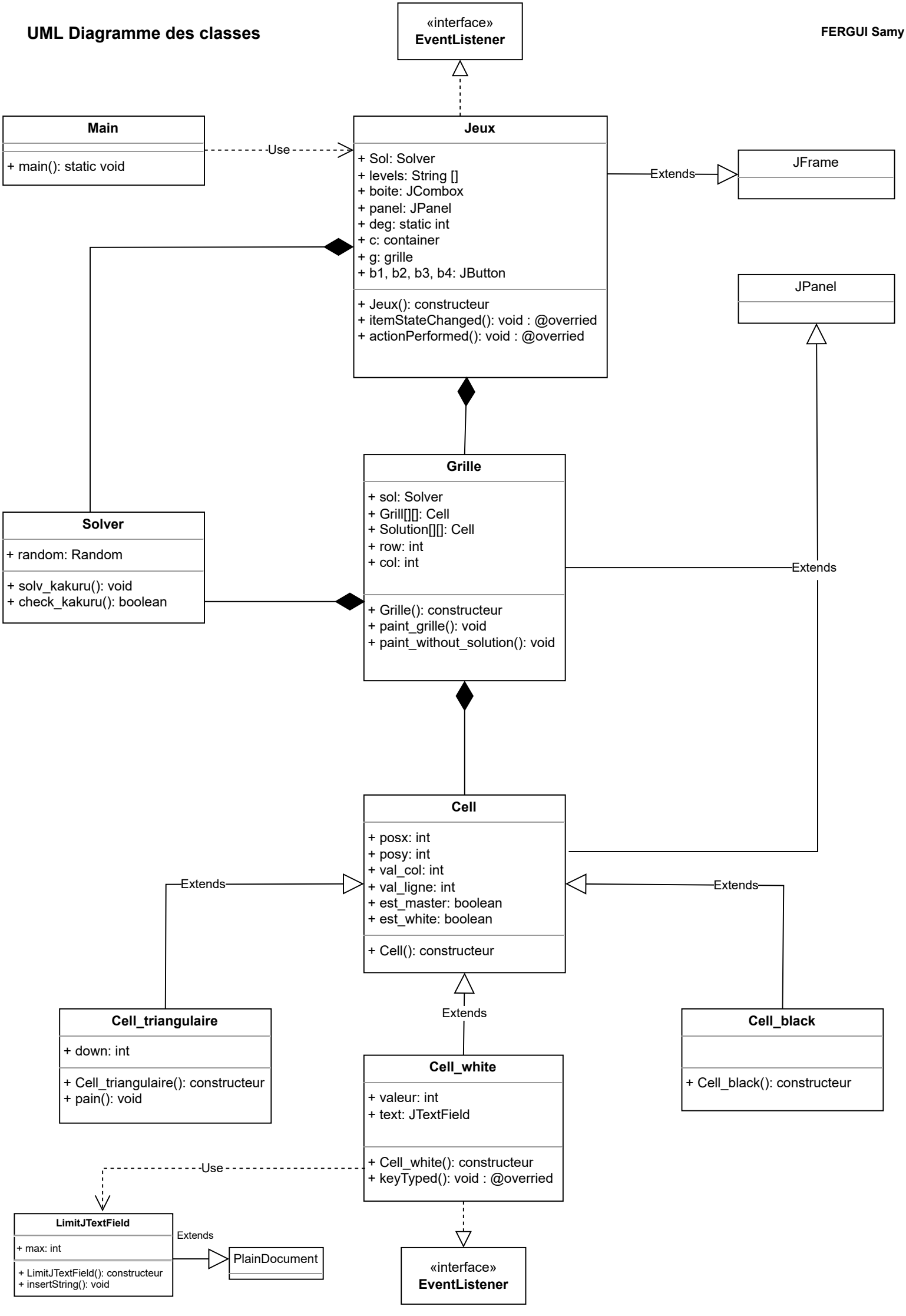
B. Checker :

Pour checker le remplissage qu'a effectué le joueur, il suffit de cliquer sur le bouton checker. c'est la solution donnée est correcte, un pop up avec un message de réussite s'affiche, sinon ce sera un message d'échec.

C. Diagramme des classes "UML" :

UML Diagramme des classes

FERGUI Samy



D. Fonctionnalitées ajoutées :

E. Niveau de jeux :

il y aura à disposition du joueur 3 niveaux, facile (grille 5*5), moyen (grille 7*7) et difficile (9*9). Le nombre total des grilles remplies avec des chiffres est 15% du nombre total des cellules. Le passage d'un niveau à un autre se fait d'une manière dynamique.

F. Hint

En cliquant sur le Hint, une recherche aléatoire se fait sur la grille ou on sauvegarde la solution de départ, quand on tombe sur une grille qui est censée être remplie par le joueur, on recopie le chiffre qui a été dans la grille avec solution dans la grille affiché au joueur et voilà.

G. Explication synthétique des algorithmes importants

1. Solv_kakuru :

notre grille avec les cellules blacks, triangulaires et white (a remplir par l'utilisateur). On passe cette grille comme paramètre à la fonction. on parcourt chaque case, quand on rencontre une cellule triangulaire, on parcours vers le bas le long de la colonne ou se trouve cette cellule, on génère à chaque fois un nombre aléatoire entre 1 et 9, avant de le mettre sur la cellule white (a remplir), on vérifie si ce chiffre n'est pas déjà utilisé en la ligne et la colonne ou se trouve la case white, si c'est le cas on met à jour le champs valeur de la cellule sinon on refait le randomize. On fait pareil avec toutes les cellules white tant qu' on atteint pas les bordures de la grille et on ne rencontre pas une autre cellule triangulaire. On refait le même processus sur le long de la ligne de la cellule triangulaire. On met à la fin à jour le champ somme colonne et somme ligne de la cellule triangulaire. On fait pareil avec toutes les cellules triangulaires du la grille. à la fin on a une grille bien remplie.

2. Check_kakuru :

On parcours les cases de la grille, qu'on tombe sur une cellule triangulaire (remplie avec des nombres) on vérifie si la somme ligne et colonne est correcte, ie la somme des chiffres des cellules se trouvant au long de la colonne (resp ligne) avant d'atteindre la prochaine cellule triangulaire ou les bordure de la grille est égal à somme colonne (resp

somme ligne) de la cellule triangulaire. si c’est le cas pour toutes les cellules de la grille on retourne vrai sinon faux.

3. Conclusion

Le projet était bel et bien une opportunité pour appliquer tous les concepts vus en POO, se familiariser avec la programmation POO en java ainsi qu'en swing, j’ai appris pas mal de choses en une semaine.
