Polytech Lyon
Departement Informatique

Projet APO

Intitulé Du Projet:

Sudoku (et variantes) : résolution et génération

Réalisé par:

- NADIR Ayoub
- MESSAOUDI Amin
- BEN MOHA Abderrahim

Année 2024-2025

Méthodologie

Notre équipe, composée de trois membres, a choisi d'adopter une approche agile afin d'assurer une conception et une implémentation efficaces de notre solution de résolution et de génération de sudokus (et variantes) en Java. Cette méthode a permis une collaboration étroite, une adaptation rapide aux imprévus et une intégration progressive des fonctionnalités.

1. Répartition des tâches et responsabilités

- **Développement du cœur algorithmique**
  Chaque membre de l'équipe a contribué à la programmation du code, avec une répartition claire des responsabilités dans le développement des fonctionnalités centrales.
- **Interface utilisateur**
  Amin s'est chargé de la conception et de l'implémentation de l'interface utilisateur.Son travail a consisté à développer une interface graphique qui permet à l'utilisateur de :
  - Choisir le type de Sudoku
  - Choisir la largeur et l'hauteur de la grille, la largeur et hauteur des blocs et la difficulté
  - La génération du Sudoku ainsi que sa résolution

- **Documentation et mise à jour des diagrammes UML**
  Ayoub a assuré la création et la mise à jour continue des diagrammes UML. Dès le début, nous avons établi des modèles comprenant :
  - **Un diagramme de cas d'utilisation**
  - **Un diagramme de classes**
  - **Un diagramme d'activités**
  - **Un diagramme d'objets**
  - **Un diagramme d'état**
  - **Un diagramme de séquence**

2. Processus de développement et communication

Pour assurer une coordination efficace nous avons adopté les pratiques suivantes :

- **Réunions régulières**
  Nous avons organisé des réunions de suivi hebdomadaires pour partager nos avancées, discuter des difficultés rencontrées. Ces points de synchronisation ont facilité l'intégration de nos contributions et renforcé la cohésion de l'équipe.

- **Utilisation de Git pour la gestion de version**
  Le code source a été géré via Git. Chaque membre travaillait sur des branches dédiées à ses modules, et nous réalisons des revues de code avant chaque fusion afin d'assurer une qualité homogène et une bonne intégration des différentes parties du projet.(Vous trouverez l'adresse de github ici: https://github.com/Zenkai04/APO.git)

3. Gestion du temps et suivi du projet

Nous avons planifié le projet en plusieurs phases :

- **Phase de conception**
  - Réflexion collective et définition des objectifs.
  - Élaboration initiale des diagrammes UML
- **Phase de développement**
  - Implémentation simultanée des modules de résolution et de génération par l'ensemble de l'équipe.
  - Développement de l'interface utilisateur par Amin.
  - Intégration progressive et tests unitaires de chaque fonctionnalité.
- **Phase d'intégration et de finalisation**
  - Validation globale du système.
  - Révision et mise à jour finale des diagrammes UML par Ayoub.
  - Préparation de la documentation finale et du rapport.

Cette planification nous a permis de respecter les échéances et d'assurer une répartition équilibrée des tâches, garantissant ainsi la réussite globale du projet.

Nos Diagrammes:

Les diagrammes sont également disponibles sur notre espace Git pour une meilleure visibilité.

Diagramme cas d'utilisation:

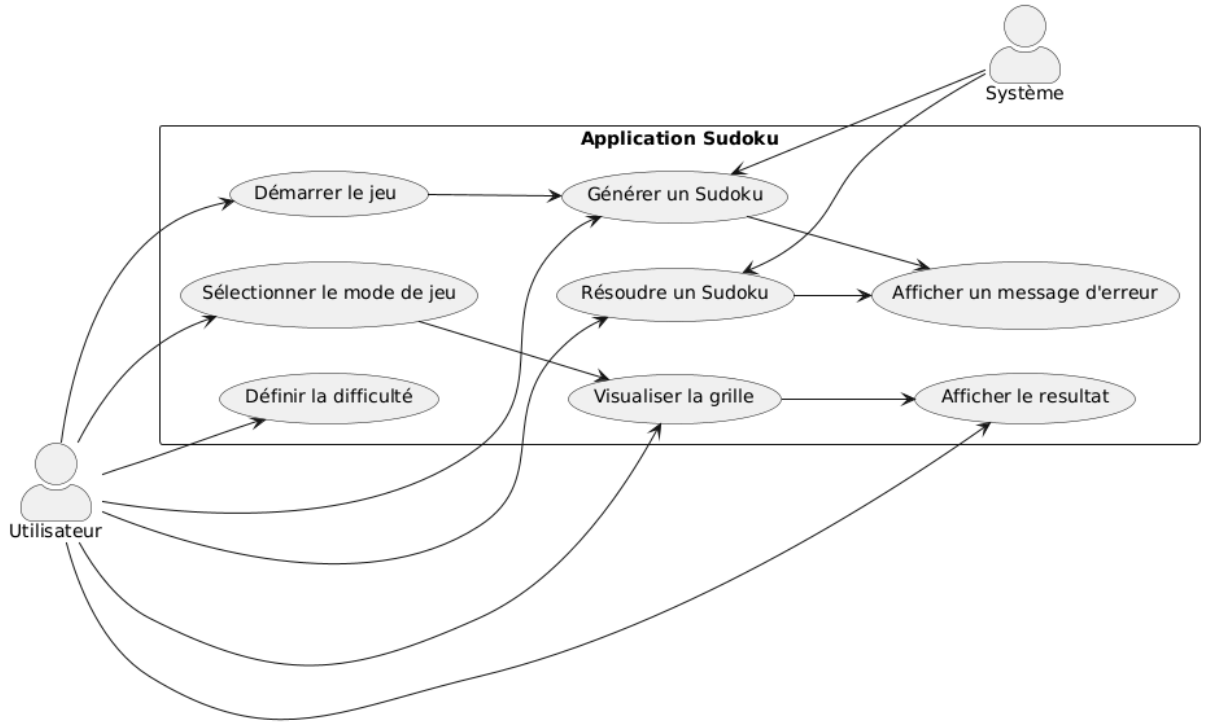


Diagramme de classe:

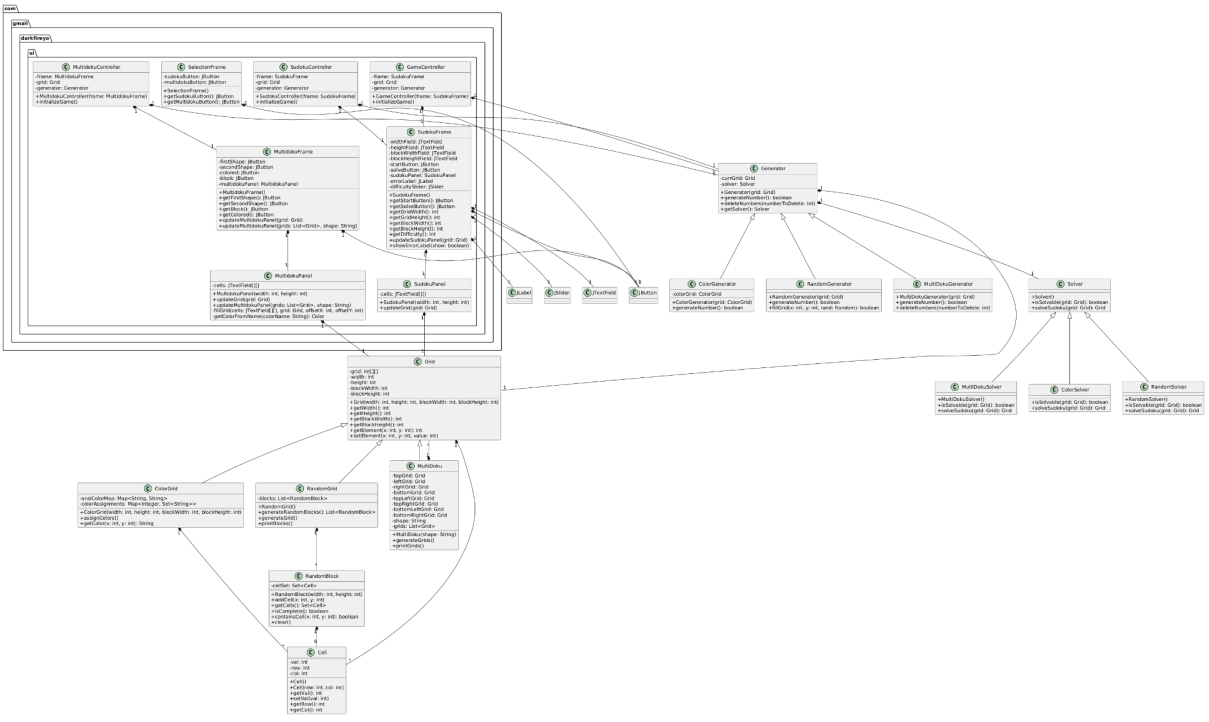
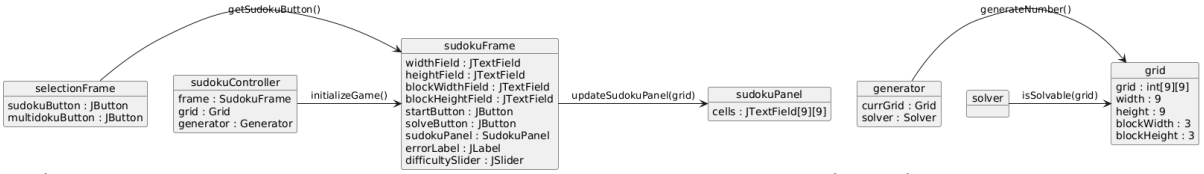
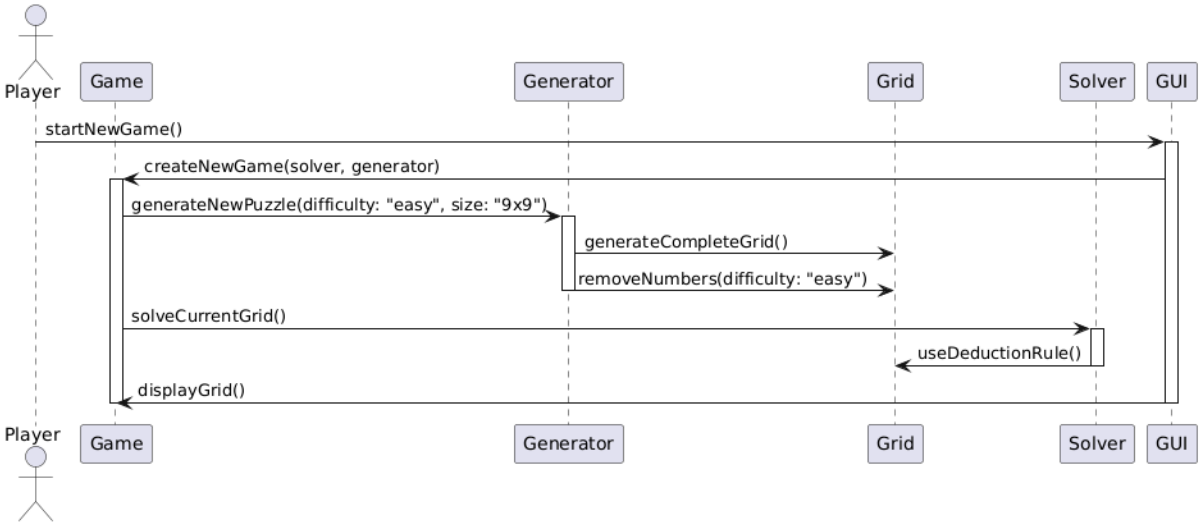


Diagramme d’objet:



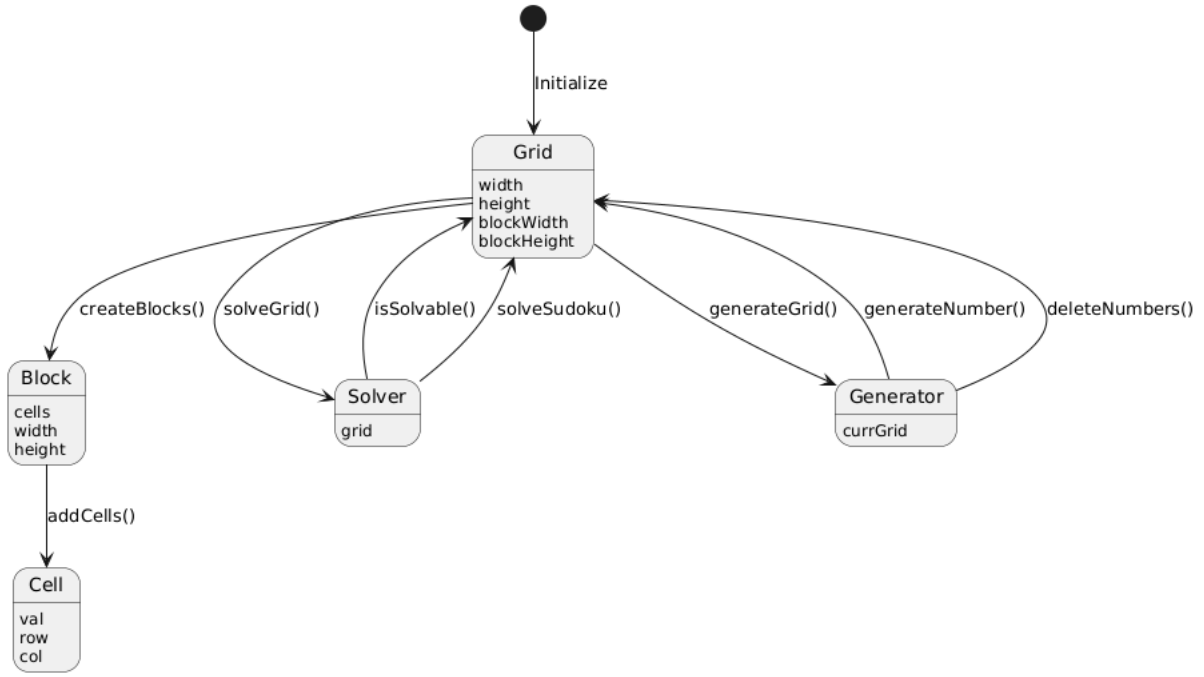
Scénario : «On lance une partie d'un sudoku de taille 9 cellules fois 9 cellules et 3 blocs fois 3 blocs à difficulté facile et sur interface graphique»

Diagramme de séquence:



Scénario : «On lance une partie d'un sudoku de taille 9 cellules fois 9 cellules et 3 blocs fois 3 blocs à difficulté facile et sur interface graphique»

Diagramme d'activité:



Difficultés rencontrés:

La plus grande difficulté que nous avons rencontré est la gestion des erreurs au cours du développement : nous avons consacré la majorité du temps passé dans ce projet à gérer les erreurs. En ce qui concerne le passage de la conception à l'implémentation, les tâches les plus compliqués se sont révélés être le MultiDoku et le Sudoku en “blocs aléatoires”: pour le MultiDoku nous avons commencé avec l'approche de définir d’abord les grilles extérieures et puis la grille centrale, cependant nous rencontrions des conflits entre les grilles et donc la grille centrale ne pouvait pas être définie. Nous avons donc opté pour d’abord définir la grille centrale et ensuite faire suivre les grilles extérieures.

Pour le Sudoku “en blocs” la tâche plus compliquée a été de définir les blocs avec des formes aléatoires : au moins le Sudoku coloré permettait d’avoir des chiffres qui n’étaient pas adjacents et appartenant au même “bloc” ou dans ce cas couleur.

Pour implémenter un Sudoku en blocs nous avons d’abord pensé de faire tout à la fois, c'est-à-dire générer un Sudoku avec des blocs “exotiques” et qui respectait la règle de “pas deux chiffres dans la même ligne ou colonne”. Pour le fait que nous avions des problèmes même pour générer un bloc nous nous sommes tournés vers une autre option : définir d’abord un sudoku qui n’avait pas deux chiffres dans la même ligne ou colonne et ensuite définir les blocs qui étaient composés par 9 chiffres distincts et adjacents ce qui posait le problème de se retrouver avec des blocs incomplets car ils étaient impossibles à compléter. Nous avons donc enfin choisi de d’abord générer les blocs aléatoires en nous assurant qu’ils ne dépassent pas les bords de la grille et qu’ils ne rentrent pas en conflit pour ensuite les adapter de sorte que dans la grille finale il n’y ait pas de violation de règle.

Avec plus de temps nous pourrions…:

Pour l’instant notre résultat final reste assez statique : nous pouvons générer des Sudoku ainsi que leur variante et nous pouvons aussi défier nos capacités en essayant de remplir une grille “classique” (sur une feuille et avec un stylo). Avec un peu plus de temps nous pourrions implémenter la saisie pour remplir directement une grille avec notre clavier, générer des dérivés du Sudoku qu’on pourrait remplir nous même au lieu de voir directement la solution.

Nous aurions également pu gérer le Sudoku avec des Blocs irréguliers en ajoutant un affichage sur la partie graphique,à la place de seulement afficher cela dans la console.

Et pour conclure, avec plus de temps nous pourrions générer des Sudokus avec plusieurs solutions.