Rapport
SAE21_2021

24/05/2022

Mathis CHAIGNEAU

Table des matières

Introduction....1
Fonctionnalités....2
Structure du programme....4
Sauvegarde d'une partie....5
Révélation en cascade....5
Conclusion....5

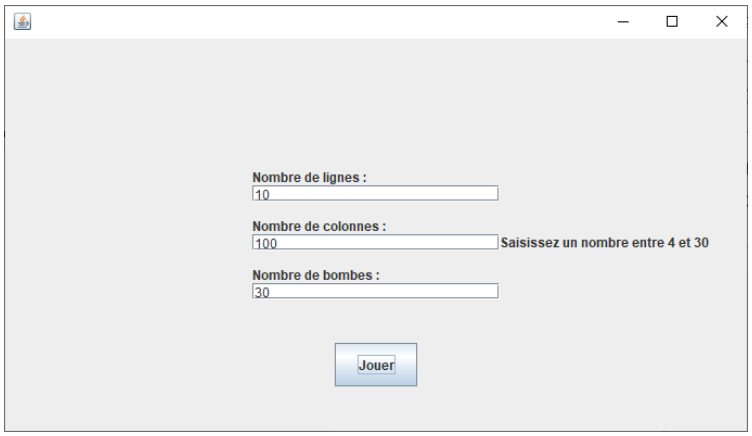
Introduction

Dans le cadre de la SAE Dev2.1, j'ai eu à réaliser un jeu de démineur. Le principe est simple, une grille d'un nombre de cases par lignes et colonnes défini par l'utilisateur, contient un nombre de bombes défini par l'utilisateur. Le but de cet utilisateur est de révéler toutes les cases non minées sans tomber sur on case minée au cours de la partie, le cas échéant, la bombe explose, la partie se termine et l'utilisateur perd.
Voici donc le rapport concernant cette SAE, qui a pour but de présenter la conception du programme ainsi que ses différentes fonctionnalités.

Fonctionnalités

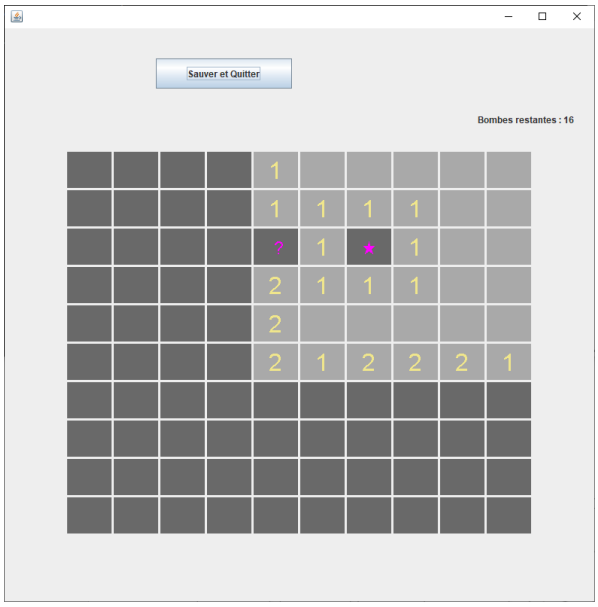
Au lancement du programme, une fenêtre contenant le menu d'accueil s'ouvre. Le menu d'accueil contient trois boutons : le bouton "Jouer" qui redirige vers le menu de choix des paramètres pour lancer une nouvelle partie, un bouton "Reprendre" qui permet de reprendre la dernière partie sauvegardée, si aucune partie n'est sauvegardée le bouton est désactivé, le bouton "Quitter" qui arrête le programme et ferme la fenêtre. Cliquer sur le bouton de fermeture dans la barre de la fenêtre a le même effet.

Le menu de choix des paramètres contient trois zones de texte titrées : la zone de texte "Nombre de lignes" pour rentrer le nombre de lignes souhaitées, la zone de texte "Nombre de colonnes" pour rentrer le nombre de colonnes souhaitées, la zone de texte "Nombre de bombes" pour rentrer le nombre de bombes souhaitées.
Ce menu contient également un bouton pour valider les choix et passer à la partie.
Le nombre de lignes et de colonnes doit être compris entre 4 et 30, le nombre de bombes doit être compris entre 1 et le nombre de cases. Dans le cas contraire, des messages d'erreur, adéquat à la valeur incorrecte, s'affiche en face de la zone de texte contenant la valeur incorrecte. Pour que l'erreur, et donc le message d'erreur qui va avec, soit prise en compte, il faut que le nombre de lignes et de colonnes rentrées sont bons.



Une fois les paramètres validés ou une sauvegarde chargée, la fenêtre s'agrandie et affiche la partie. La disposition d'une partie est composée : d'un bouton "Sauver et quitter" en haut, celui-ci permet de sauvegarder la partie en cours et d'arrêter le programme, un texte "Bombes

restantes" en haut à droite, celui-ci affiche le nombre de bombes que comporte la grille moins le nombre de marqueurs de supposition présents sur les cases de la grille, la grille au milieu, celle-ci contient les cases à révéler et à déminer. Faire un clic droit sur l'une de ces cases : ajoute le marqueur de suspicion si aucun marqueur n'était présent, ajoute le marqueur de supposition si le marqueur de suspicion était présent, retire le marqueur de supposition si ce dernier était présent. Faire un clic gauche sur une de ces cases : la révèle si aucun marqueur n'était présent, enlève le marqueur s'il y en avait un.



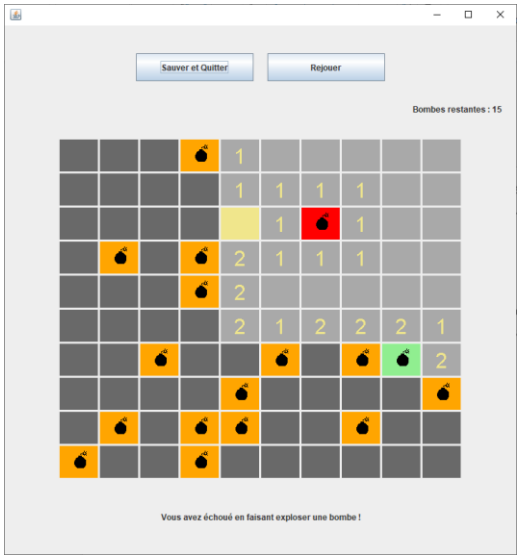

Une case sans marqueur


Une case avec le marqueur de suspicion


Une case avec le marqueur de supposition

Révéler une case affiche le nombre de case minées autour d'elle, s'il n'y en a pas, rien n'est afficher. Les cases aux alentours n'ont pas non plus de case minée autour d'elles, elles sont révélées et ainsi de suite.
Cliquer sur une case minée révèle la grille en mode défaite. Révéler toutes les cases non minées révèle la grille en mode victoire.

Terminer une partie, que ce soit par une défaite ou par une victoire, fait apparaître le message adapté en bas de la fenêtre et fait apparaître le bouton "Rejouer" qui renvoie vers le menu de choix des paramètres pour lancer une nouvelle partie.



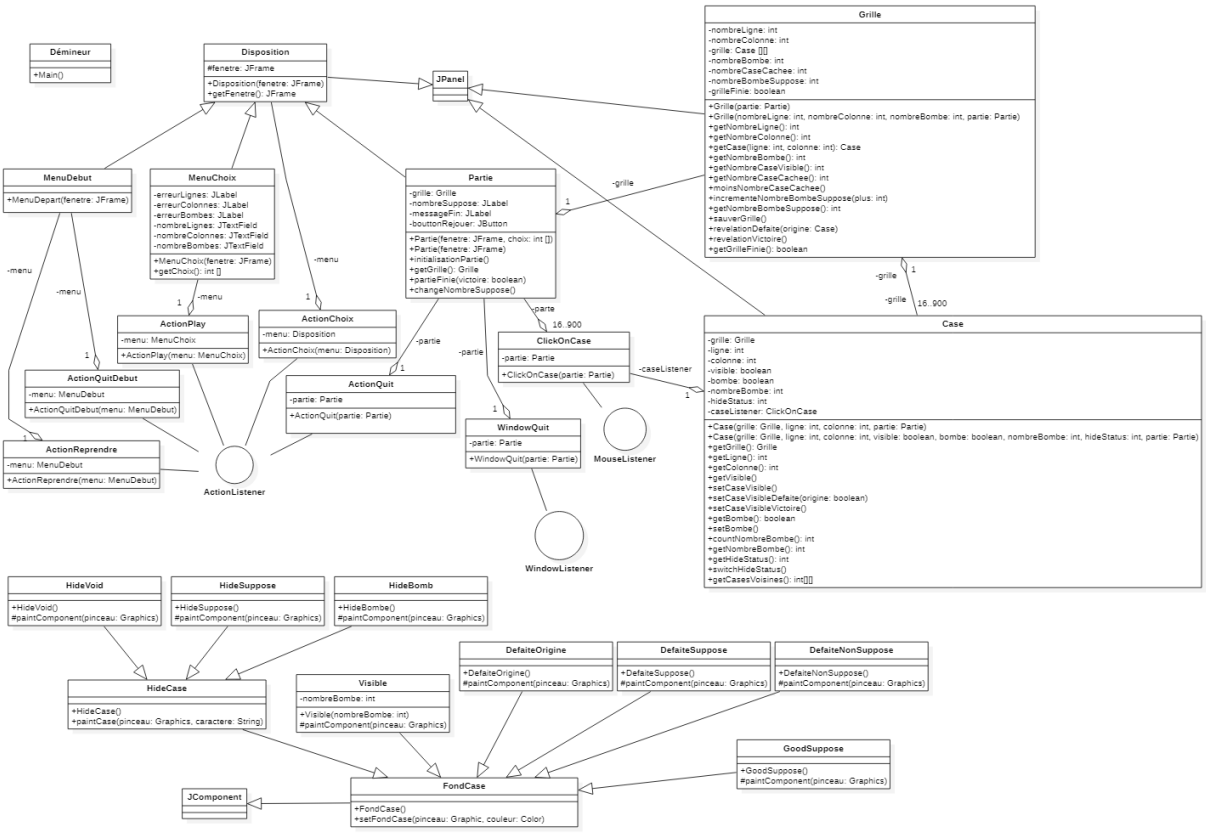

Un marqueur de supposition sur une case minée ou une case minée en cas de victoire


Pas de marqueur de supposition sur une case minée en cas de défaite


La case minée à l'origine de la défaite

Un marqueur de supposition sur une case non minée

Structure du programme



Voici le diagramme de classes du projet, le fichier source est joint aux sources du projet.

Les deux classes les plus importantes du projet sont la classe Grille et la classe Case. C'est dans ces classes, ainsi que dans la classe ClickOnCase, qui est un listener permettant de faire des cases des observateurs, que le principal de l'algorithmique se trouve.

D'un point de vue graphique, le projet est décomposé en trois parties/classes : le menu d'accueil, c'est la classe MenuDebut, le menu de choix des paramètres pour le début d'une partie, c'est la classe MenuChoix, la partie, dans laquelle l'utilisateur joue et qui contient donc la grille, c'est la classe Partie. Ces trois classes sont des classes filles de la classe Disposition, qui est une base pour ces trois parties graphiques, qui est elle-même une classe fille de la classe JPanel, cela permet d'ajouter ces différentes parties graphiques plus simplement dans la fenêtre.

Pour la partie graphique des cases, il y a une classe pour chaque état possible. La classe Case est une classe fille de JPanel et les classes graphiques sont des classe filles de la classe FondCase, qui est une base pour le graphique des cases, qui est elle-même une classe fille de la classe JComponent. Cela permet de changer l'état graphique d'une case très simplement en supprimant son ancien état et en ajoutant une nouvelle instance de la classe graphique souhaitée.

Sauvegarde d'une partie

La fonctionnalité de sauvegarde est la méthode sauverGrille de la classe Grille. Cette méthode ne prend pas d'argument et ne renvoie rien. Elle ne sauvegarde, dans un fichier save.bin, en fait que certaines variables de la classe Grille suivies de quelque variable de la classe Case, juste assez pour pouvoir revenir à l'état sauvegardé. Quand l'on souhaite sauvegarder une partie, c'est donc cette méthode qui est appelée. Elle écrit alors dans le fichier un boolean qui sert à savoir si la sauvegarde présente dans le fichier doit être chargée, autrement dit, si c'est une sauvegarde expirée ou non. Elle écrit ensuite quatre int qui correspondent au nombre : de lignes, de colonnes, de bombes, de cases cachées. Pour finir, elle écrit autant de fois qu'il y a de cases : un boolean représentant la visibilité de la case, un boolean pour savoir si la case est minée, un int représentant le nombre de bombes autour de la case, un int représentant le marqueur présent sur la case.

Pour charger une partie, un constructeur est prévu dans la classe Grille. Il ne prend qu'un objet de la classe Partie comme argument. Il lit les variables appartenant à la grille puis les variables faisant référence à des cases. Un constructeur qui prend plus d'arguments est défini dans la classe Case et utilisé par le constructeur de la classe Grille.

Révélation en cascade

Au moment de révéler une case, un compte du nombre de cases minées autour de la case est fait par la méthode countNombreBombes. Pour aider les méthodes countNombreBombes et getNombreBombes, la méthode getCasesVoisines est appelée pour obtenir les cases voisines à la case. Si le nombre de cases minées autour de la case est égal à zéro alors on appelle la méthode setCaseVisible de toutes les cases voisines non révélées. Ainsi Il n'y a pas besoin de gérer la possibilité qu'il y ait beaucoup de cases à révéler, il y a un effet en cascade.

Conclusion

Avant la décomposition du projet, via la création du diagramme de classes, le projet me semblait vraiment difficile, surtout du point de vue graphique. Commencer par créer le diagramme de classe m'a permis d'avoir une idée plus précise de ce que j'allait devoir faire et de comment le faire.
Après la finalisation d'une première version du diagramme de classe, j'ai trouvé le projet assez simple finalement. Dans la réalisation du projet a continué dans cette lignée et je n'ai eu que peu d'embuches. Les principales ont été graphiques, au niveau de la disposition des éléments dans la fenêtre. Pour cela j'ai passé beaucoup de temps à chercher différents "Layout Manager" pour trouver celui qui me permettrait facilement de disposer les différents éléments dans la fenêtre comme je le voulais. Un deuxième point sur lequel j'ai passé beaucoup de temps, et cette fois sans succès, est la redirection automatique vers le menu après un court temps après la fin d'une partie. Cette fonctionnalité a donc été remplacée par le bouton "Rejouer" qui s'affiche en fin de partie.

Ce projet m'a permis d'approfondir mon expérience dans la recherche d'information, que ce soit dans la documentation officielle ou dans d'autres sources et d'approfondir ma connaissance et ma compréhension du langage java.