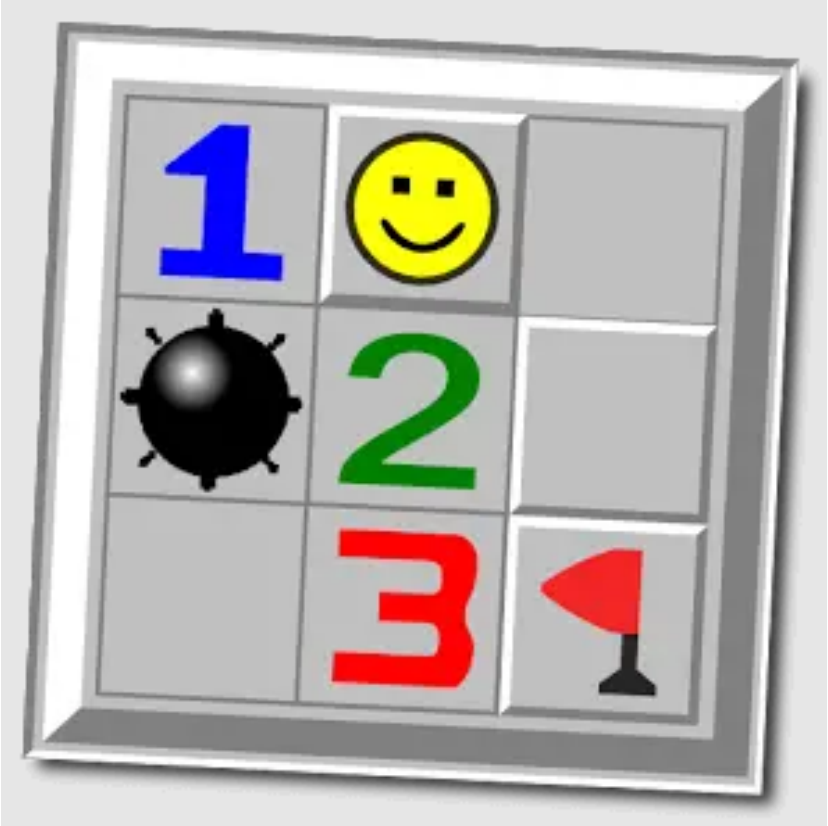
Mini-Projet : jeu de démineur



Koccou MINAGNIKPO

AVRIL 2023

Table des matières

1 Introduction

Le démineur est un célèbre jeu de réflexion et de stratégie qui consiste à déminer un champ de mines tout en évitant de les faire exploser. Dans le cadre de ce mini projet de Java, j'ai développé une version du démineur en utilisant les principes de la programmation orientée objet. Ce jeu offre une interface graphique intuitive et une expérience de jeu interactive qui permet aux utilisateurs de jouer au démineur en temps réel. Le rapport qui suit présentera une vue d'ensemble de cette approche de développement, des fonctionnalités clés et des défis rencontrés tout au long de l'elaboration du code .

2 Explication de la démarche élaborée

Ce code est une implémentation en Java du célèbre jeu de démineur. Le jeu se joue sur une grille de cases carrées, certaines contenant une mine et d'autres non. Le but du jeu est de découvrir toutes les cases qui ne contiennent pas de mines sans jamais tomber sur une case qui en contient une.

La classe Grille est utilisée pour représenter la grille du jeu. Elle contient des tableaux pour stocker l'état des cases de la grille, des méthodes pour initialiser la grille avec un certain nombre de mines, placer les mines de manière aléatoire, découvrir une case, marquer une case et afficher la grille.

La méthode initialiser initialise tous les tableaux avec des valeurs par défaut. La méthode **placerMines()** est appelée pour placer les mines aléatoirement sur la grille. Un objet Random est créé pour générer des nombres aléatoires. Tant que toutes les mines n'ont pas été placées, un nombre aléatoire est généré pour la position de la mine. Si cette position ne contient pas déjà une mine, la mine est placée à cet endroit. Le nombre de mines voisines de chaque case est également mis à jour en parcourant les cases adjacentes à la mine

La méthode **decouvrirCase** est appelée lorsque l'utilisateur clique sur une case pour la découvrir. Si la case contient une mine, le jeu est perdu. Sinon, la case est découverte et si elle n'a pas de voisins minés, toutes les cases adjacentes sont découvertes également. La méthode **afficher** affiche la grille avec l'état actuel de chaque case, en utilisant des boutons JButton pour représenter chaque case. Si la case n'a pas encore été découverte, le bouton est vide. Si la case a été marquée par l'utilisateur comme contenant une mine, le bouton affiche un "M". Si la case contient une mine, le bouton affiche un "X". Sinon, le bouton affiche le nombre de mines voisines. La méthode **afficherToutes** est appelée lorsque le jeu est perdu avec un message Vous avez perdu ! La partie est finie, Partie terminée" et affiche toutes les cases de la grille, avec leur contenu révélé.

La méthode **marquerCase** est appelée lorsque l'utilisateur marque une case comme contenant une mine. La méthode **estTerminee** vérifie si toutes les cases

sans mine ont été découvertes et si toutes les cases contenant une mine ont été marquées comme telles. Si c'est le cas, le jeu est gagné avec un message "Félicitations, vous avez gagné !", "Partie terminée".

3 Déroulement du jeu

Dans cette partie, je vais vous présenter le déroulement de ce jeu.

Tout d'abord, l'utilisateur sera invité à saisir les dimensions de la grille (largeur et hauteur) ainsi que le nombre de mines qui y seront cachées. Ensuite, la grille s'affichera avec une interface graphique permettant à l'utilisateur de jouer. Des choix lui seront proposés tout au long du jeu.

- En saisissant la lettre "m", il pourra marquer une case avec **M** en saisissant ses coordonnées (largeur, hauteur).
- En utilisant la lettre "s", il pourra sauvegarder l'état actuel de la grille dans un **fichier.txt** qu'il saisira grace a la **methode save**
- avec la lettre "c", il pourra charger l'état précédemment sauvegardé grace a la **méthode load** qui va appeler le fichier dans lequel a été sauvegardé l'état de la grille.

Le jeu prend fin lorsque toutes les cases non minées ont été découvertes et que les cases potentiellement minées ont été marquées. Cela implique que le joueur a gagné la partie. En revanche, si le joueur tombe sur une case minée, le jeu prend fin et toutes les cases sont révélées.

4 Les choix architecturaux

Voici une description de certains choix architecturaux qui ont été faits dans ce code :

Utilisation d'un modèle de grille pour stocker les informations sur le jeu : Le code utilise un modèle de grille pour stocker les informations sur le jeu. Les données sont stockées dans des tableaux à deux dimensions. Cela permet de stocker facilement l'état de chaque case de la grille, tel que la présence ou non d'une mine, le nombre de mines voisines, si la case a été découverte ou non, si elle est marquée comme une mine, etc. Cette approche rend également plus facile la manipulation et la mise à jour des données, ainsi que l'implémentation de différentes fonctionnalités du jeu.

Utilisation d'une interface graphique pour afficher la grille : Le jeu utilise une interface graphique pour afficher la grille. Cela permet une expérience utilisateur plus agréable en fournissant une interface utilisateur conviviale pour jouer le jeu. L'interface graphique est créée en utilisant la bibliothèque Swing

de Java, qui fournit une grande variété de composants d'interface utilisateur.

Utilisation de la récursion pour la découverte des cases vides : Lorsqu'une case vide est découverte, le code utilise la récursion pour découvrir toutes les cases vides voisines. Cette approche est utilisée car elle est plus simple et plus élégante que d'utiliser une boucle pour parcourir toutes les cases voisines à la recherche de cases vides. Cela permet également une meilleure séparation des préoccupations dans le code, car la fonction de découverte de case est entièrement responsable de la découverte de cases vides, tandis que la fonction d'affichage de la grille est responsable de l'affichage de la grille.

Utilisation de boutons pour représenter les cases de la grille : Le jeu utilise des boutons pour représenter les cases de la grille. Cette approche permet d'utiliser facilement la gestion des événements pour déclencher des actions lorsqu'un bouton est cliqué. De plus, cela facilite également la mise à jour des boutons lorsqu'une case est découverte ou marquée comme une mine.

5 Conclusion

En résumé, dans ce mini projet j'ai implémenté le jeu de démineur en java permettant à un utilisateur d'y jouer avec une interface graphique pour une meilleur expérience,le défi qui lui sera proposé est de découvrir toutes les cases non mièńées et pouvoir marquer les cases susceptibles de contenir des mines,sans pour autant tombé sur une case miné sinon la partie est perdue.