Démineur en java

Chassefeyre Alexandre, Boyer Yoan

05/04/23

Table des matières

1 Introduction

Le but de ce projet est de réaliser le jeu du démineur en java. Nous avons réalisé 2 versions : Une qui se joue directement dans la console et une qui se joue avec une interface graphique. L'utilisateur choisi quelle version il souhaite utiliser au lancement du programme.

2 Diagramme UML

Nous avons créé une classe *Game* qui permet d'interagir avec l'utilisateur dans le terminal en demandant les propriétés de la grille ou la case qui doit être découverte ou marquée. Les classes *GUI* et *MouseClick* permettent les mêmes fonctionnalités pour l'interface graphique. La classe *Grid* gère les propriétés du plateau et la classe *Square* celle des différentes cases. Les enum SquareState, GameState et Direction permettent d'avoir des états différents pour les cases (marquées, découvertes ou cachées) et pour le jeu (en cours, gagné, perdu ou sauvegardé) ainsi qu'une direction pour chaque voisin d'une case.

3 Choix du stockage des données

Pour réaliser la grille de jeu nous avons décidé de stocker l'ensemble des cases dans un tableau à 2 dimensions. Cela permet par la suite de faciliter le parcours de la grille de jeu et d'accéder plus facilement à une case en particulier.

Cependant, pour faciliter la méthode récursive qui permet de découvrir les cases (détaillée ci-dessous), nous avons aussi décidé de stocker tous les cases adjacentes à une case dans une *EnumMap*. Cette *EnumMap* prend comme clé toutes les valeurs possibles de l'énumération *Direction* et lui associe la case adjacente correspondante si elle en a une, sinon *None*. Durant la création de la grille, chaque case créée est donc reliée à ses voisines déjà existantes et inversement.

4 Méthode récursive *Uncover*

Cette méthode permet de découvrir toutes les cases voisines à celle sélectionnée tant qu'il n'y a pas de bombes autour d'elles. Elle va parcourir toutes les valeurs non-nulles de l'*EnumMap* stockant les voisins de la case et va rappeler la fonction pour chacune d'entre elles.

5 Méthodes de sauvegarde et rechargement

Pour enregistrer l'état courant de la grille, la méthode *save(path)* de la classe *Grille* va écrire dans un fichier les informations actuelles de la partie. Les deux premières lignes du fichier représentent la largeur et la longueur du tableau et les lignes suivantes vont représenter les cases avec un chiffre différent selon leur état :

1. Case découverte
2. Case cachée étant une mine
3. Case cachée n'étant pas une mine
4. Case marquée étant une mine
5. Case marquée n'étant pas une mine

Ainsi, pour recharger une partie enregistrée, la méthode *load(path)* va lire le fichier et récupérer la taille de la grille. Les caractères du fichier représentant les cases vont ensuite être lus un par un et les cases vont retrouver leur état en fonction du chiffre. On va donc rappeler la méthode qui définie les voisins pour chaque case puisque ces derniers ne sont pas stockés dans le fichier. Enfin, la méthode *findMines()* va permettre de retrouver les mines dans le plateau qui a été rechargé.

6 Interface graphique avec Swing

Nous avons commencé par représenter le plateau dans le terminal mais nous avons ensuite décidé de créer aussi une interface graphique en utilisant la bibliothèque standard *Swing*. La classe *Game* va permettre de créer l'affichage de la fenêtre avec des JPanel et une tableau de JButton qui correspondent tous à une case du plateau. Pour gérer les cliques gauche et droit de la souris, nous avons créé la classe *MouseClick* qui hérite de *MouseAdapter* et qui permet de

changer l'apparence des boutons en fonction du clic et de l'état de la case correspondante. Pour faciliter la modification des boutons, chaque case possède un attribut button qui peut être modifié par les méthodes *uncoverButton*, *markButton* et *mineButton*.