UPEC
Connaissance - Action

UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

Rapport Sae 2.1 Dev

Rapport du démimneur

Fait par

Nicolas Petronis
Yuness Soussi

IUT sénart-Fontainebleau

Fait le 22/05/2022
nicolas.petronis@etu.u-pec.fr
yuness.soussi@etu.u-pec.fr

Yuness Soussi
Nicolas Petronis

Sommaire

Yuness Soussi
Nicolas Petronis

Introduction

Pour ce micro-projet il nous a été demandé de réaliser un jeu du Démineur dans le langage informatique Java. Le démineur traditionnel se joue seul et est composé d'une grille de cases allant d'une plage de 4 à 30 lignes et colonnes(Si le démineur a 4 lignes et 5 colonnes, alors il y a 20 cases) .
Le but est de découvrir toutes les cases du plateau sans tomber sur une mine.

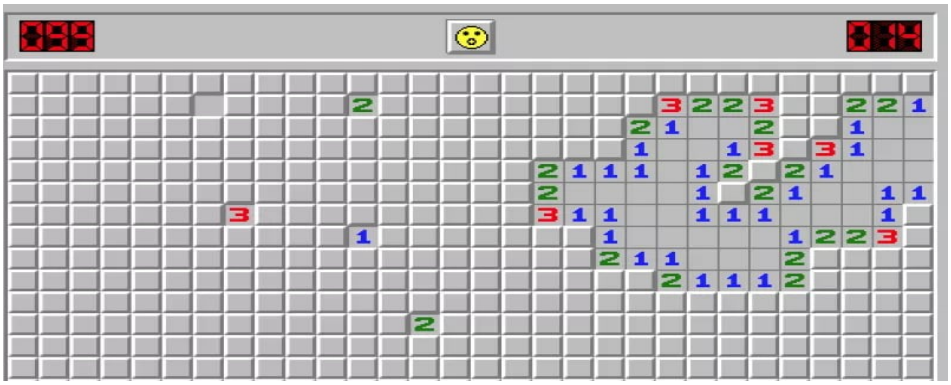
Les mines ainsi que le nombre de lignes et colonnes, sont sélectionnées par le joueur, ainsi le joueur peut configurer une partie pour qu'elle soit adéquate à son niveau.
Au démarrage de la partie après configuration du jeu, le joueur doit pouvoir cliquer sur une case pour la découvrir. Si celle-ci ne contient aucune bombe, alors elle devient vierge (et d'une couleur différente que les cases non découvertes) ou alors elle se remplit par un 0 et toutes les cases qui y sont adjacentes sont aussi découvertes (peut provoquer une réaction en chaîne). Si la case est voisine d'une case contenant une bombe alors elle doit afficher en son sein le nombre de cases voisines contenant une bombe.
Si le joueur clique sur une case contenant une bombe alors il doit être notifié qu'il a perdu et il ne peut plus continuer la partie.

Le sujet nous impose de permettre au joueur de laisser des repères sur la grille pour l'aider à finir le Démineur, ainsi si le joueur fait un premier clic droit sur une case non découverte, la case doit afficher un «?» notifiant ainsi que le joueur à doute concernant le fait que la case soit une bombe. Si le joueur refait un clic droit sur la case notifiée d'un point d'interrogation alors celle-ci affiche un nouveau symbole (dans notre cas un drapeau) notifiant que le joueur est certain qu'il y a une bombe.

Le joueur peut en cours de partie sauvegarder cette partie et quitter pour la reprendre plus tard. Le nombre de mines moins le nombres de repères doit aussi être indiqué à l'écran pour que le joueur sache le nombre de mines qu'il lui reste.

Le joueur gagne la partie lorsqu'il a découvert toutes les cases ; étant donné que la disposition des mines se fait de manière aléatoire, si le joueur choisit peu de mines et une grande grille, le joueur peut terminer la partie en un clic (réaction en chaîne).

Démineur classique de Windows :



Yuness Soussi
Nicolas Petronis

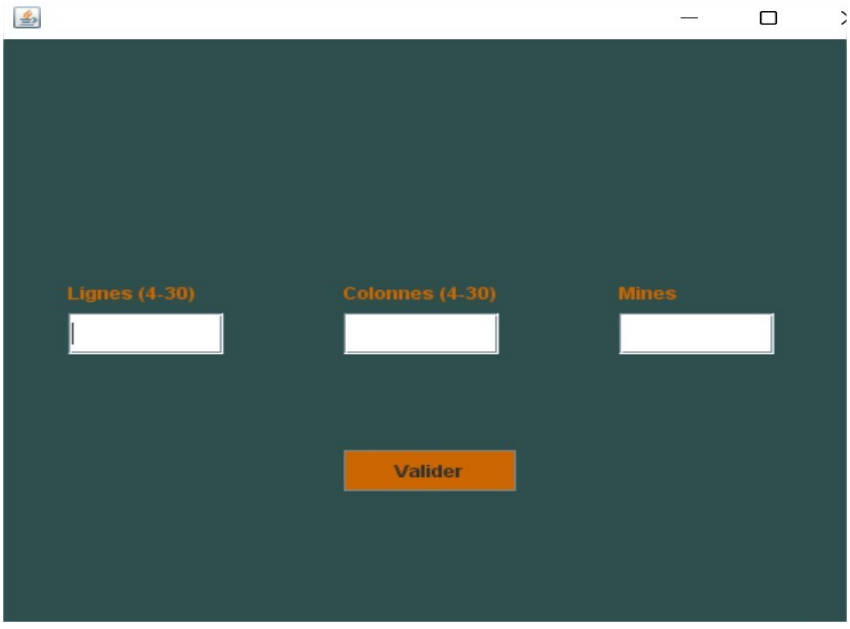
Description des fonctionnalités

Nous allons vous présenter toutes les fenêtres de notre Démineur ainsi que les fonctionnalités qui y sont présentes.

Menu:

Dans le menu du jeu, 3 fonctionnalités sont présentes. Le joueur peut cliquer sur « Nouvelle partie » pour atterir sur la fenetre des parametres et créer une nouvelle partie. Il peut aussi cliquer sur « Reprendre » pour continuer la dernière partie de démineur si celle-ci ne s'est pas finie. Cependant si aucune partie n'est en cours, l'utilisateur ne pourra pas cliquer sur reprendre, et pour finir il peut cliquer sur « Quitter » pour quitter le Démineur en fermant la fenêtre.

Configuration de la partie:



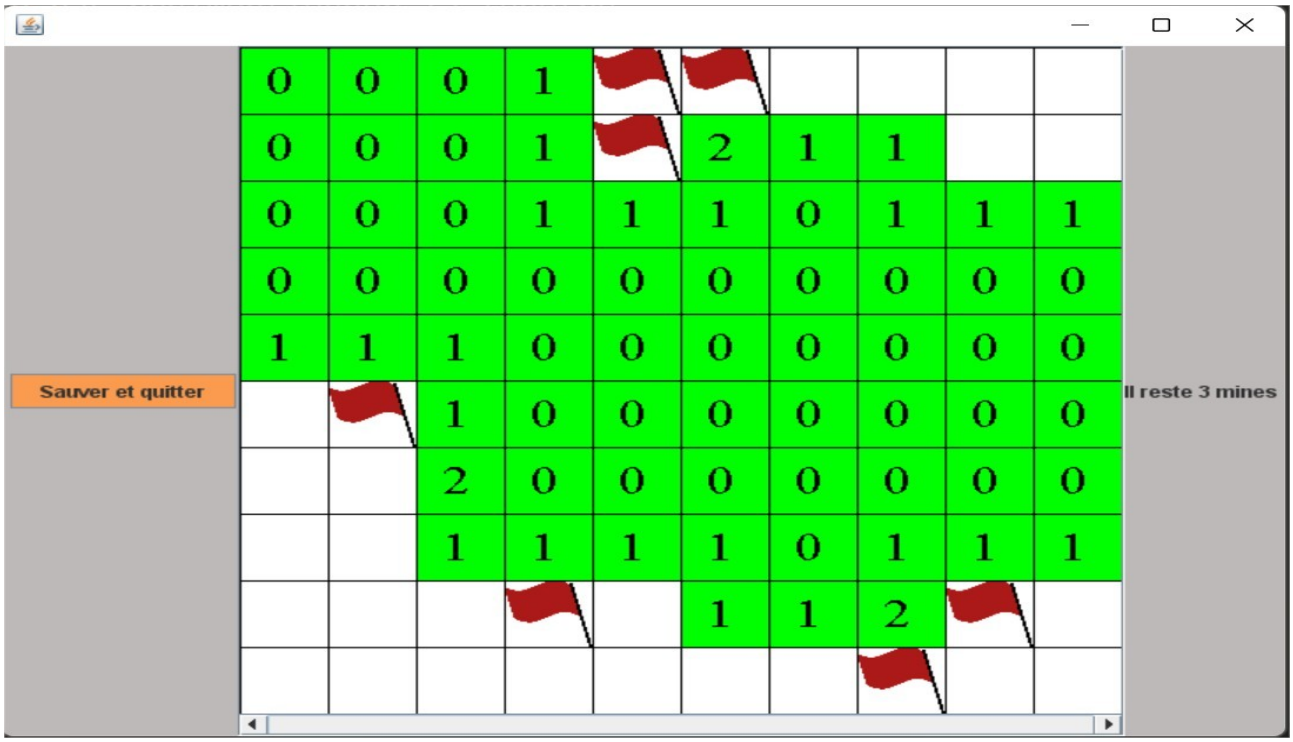
Dans cette fenêtre le joueur peut entrer le nombre de lignes, de colonnes et de bombes qu'il souhaite dans sa partie de Démineur.

Toutefois il y a une limite, il est autorisé à entrer seulement entre 4 et 30 lignes et colonnes.

Concernant le nombres de bombes, il ne peut pas entrer plus de mines qu'il n'y a de lignes et de colonnes. Dans le cas contraire, il sera bloqué et devra donc entrer un nombre de mines inférieurs à son premier choix de nombre de lignes et de colonnes pour pouvoir cliquer sur valider.

Nonobstant il n'y a aucune limite minimale pour le nombre de bombes. Ainsi le joueur peut décider de mettre 0 bombes pour avoir un sentiment de satisfaction en gagnant une partie… :)

Partie :

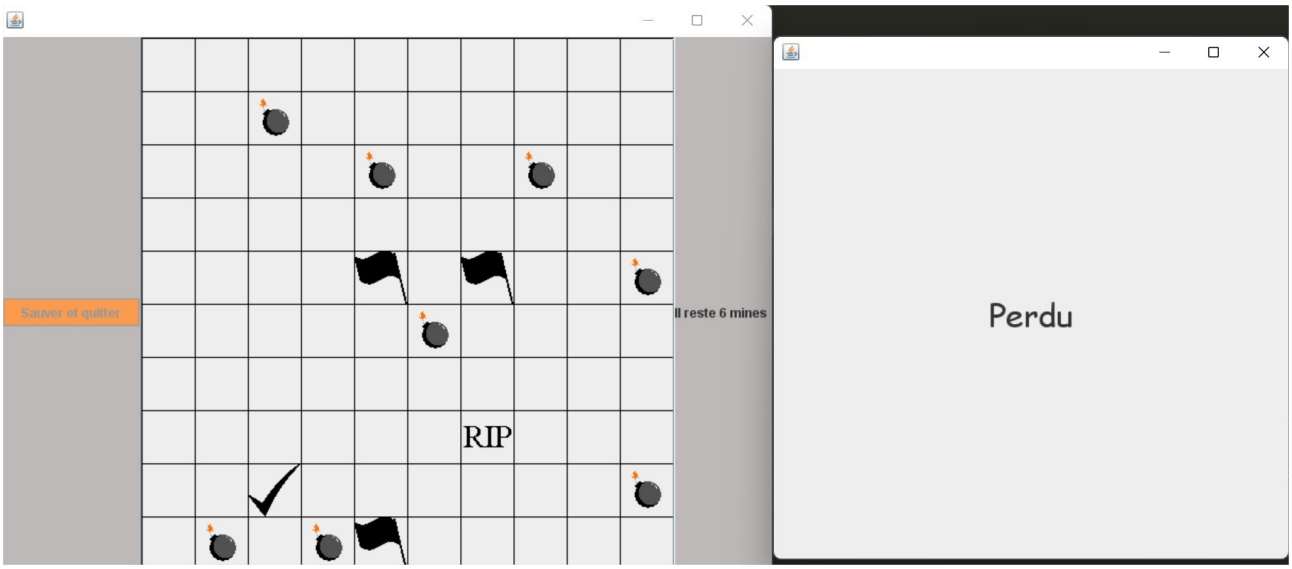


Dans cette fenêtre vous pouvez voir une partie en cours, voici la liste des actions qu'un joueur peut effectuer dans une partie:

- Cliquer sur une case pour la découvrir (clic gauche)
- Cliquer sur une case pour poser un drapeau (clic droit)
- Cliquer sur sauvegarder et quitter, pour quitter la partie en cours tout en la sauvegardant pour ensuite la reprendre depuis le menu.

Evidemment, si le joueur a perdu ou gagné, il ne peut pas cliquer sur « Sauver et quitter », dans le cas contraire le joueur pourrait reprendre la partie juste avant d'avoir perdu, ce qui serait considéré comme de la triche.

Perdu :



Si le joueur a perdu, une fenetre « PERDU » s'affiche à l'écran. Derriere cette fenetre il y a la partit actuelle du joueur, avec l'emplacement de toute les mines qui s'affiche, l'emplacement de tout les drapeaux, les endroits ou il y a eu un drapeau et une mine caractérisé par le « check », ainsi que l'endroit ou le joueur est mort.
Le bouton « sauver et quitter » devient inutilisable.

Gagné :

Si le joueur a gagné, une fenetre gagné s'affiche à l'écran. Le bouton « sauver et quitter » devient inutilisable.

En dehors de tout ça, le joueur ne peut rien faire d'autre durant une partie de Démineur. Il peut bien sûr quitter le Démineur sans sauvegarder, en quittant par le bouton croix de la fenêtre ou tout simplement déplacer le Démineur, mais 0 part ces options basiques rien d'autre n'est possible.

Structure du programmes

Notre Démineur a été totalement codé en Java et se comporte de 12 fichiers Java, 4 images au format png et un makefile pour compiler rapidement le programme.

Ci-dessous, la liste des fichiers Java avec leurs fonctionnalités :

Menu.java → Permet d'afficher le menu de départ.

Parametre.java → Permet à l'utilisateur de sélectionner le nombre de mines, lignes et colonnes souhaité

VueGrille.java → Permet d'afficher la grille de jeu

Partie.java → Fichier contenant la classe Partie qui est la classe maîtresse du programme ; c'est au sein de la classe Partie qu'a lieu la modélisation d'une partie (grille, case, nb de mines…)

Grille.java → Permet de modéliser la grille de jeu

FenetrePerdu.java → Permet d'afficher la fenêtre perdue

FenetrePartie.java → Permet d'afficher la partie

FenetreGagne.java → Permet d'afficher la fenêtre gagner

Controleur.java → Permet de faire la liaison entre l'objet Partie et la vue FenetrePartie.java

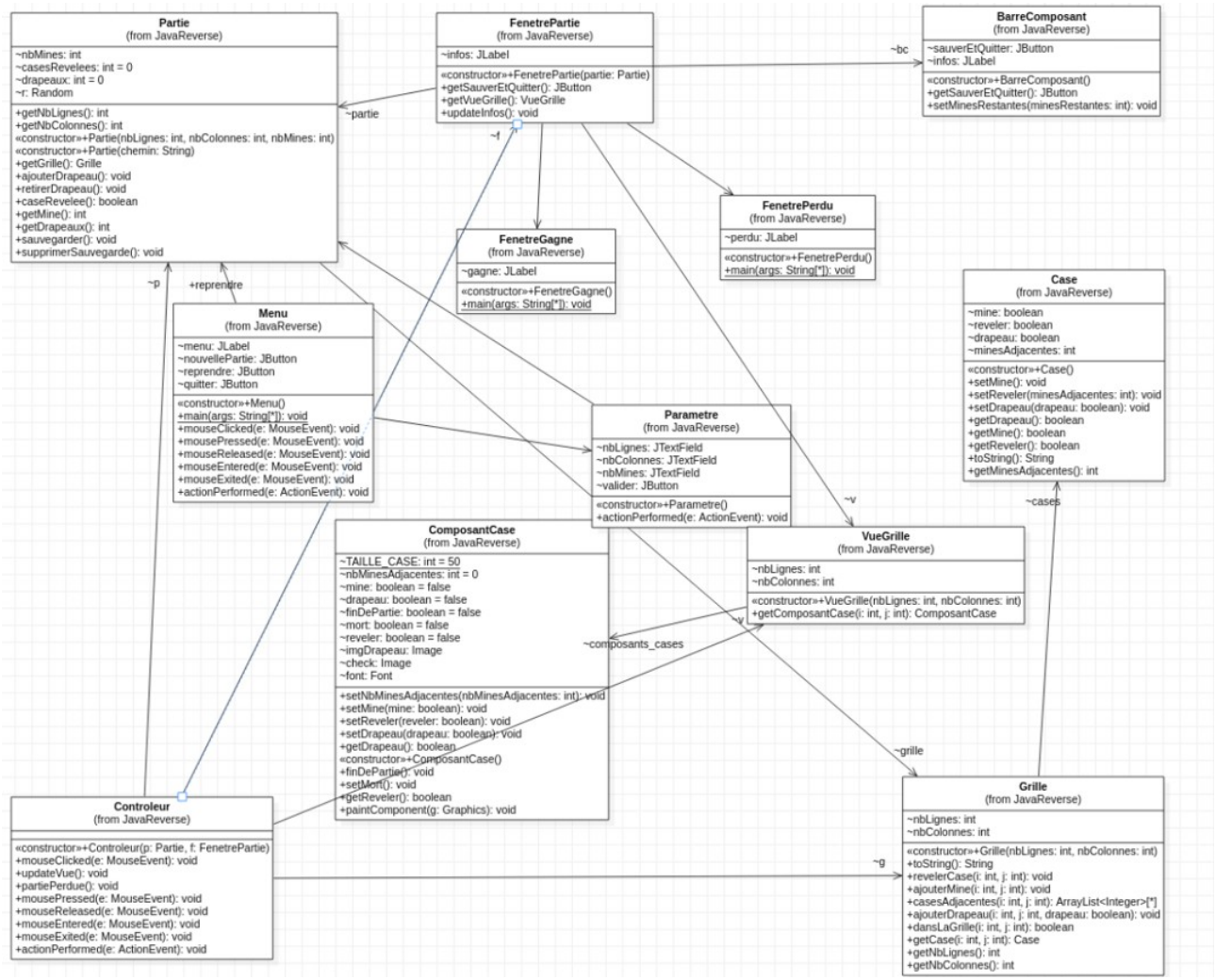
ComposantCase.java → Permet d'afficher les cases.

Case.java → Permet de modéliser une case de la grille.

BarreComposant.java → Permet d'afficher le bouton pour quitter et sauvegarder une partie ainsi que le nombre bombes restant

Voici un diagramme de classe pour mieux visualiser le fonctionnement global de notre programme ; le diagramme a été réalisé avec StarUML

Yuness Soussi
Nicolas Petronis



Explication du mécanisme de sauvegarde

Lorsque le joueur clique sur sauvegarder et quitter, un fichier fichier.txt se crée et contient les diverses informations de la partie en cours, l'emplacement des mines, l'emplacement des cases révélées, des cases avec un drapeau et le nombre de lignes et colonnes y sont inscrit.

C'est la méthode sauvegarder qui se charge de la création du fichier et de l'incrémentation des valeurs de la partie en cours.

C'est en revanche dans le constructeur de la classe Partie que le programme va charger les données du fichier fichier.txt pour reprendre la partie là où elle s'était terminée.

Lorsque l'utilisateur finit une partie, qu'il ait perdu ou gagné, la fonction supprimerPartie est appelée pour supprimer le fichier fichier.txt, ainsi sans fichier le joueur ne peut pas reprendre la partie, le bouton reprendre sur le menu devient donc inutile et ne lance rien.

Explication de l’algorithme permettant de révéler plusieurs cases

La fonction revelerCase prend en paramètre la position i et j d'une case.
Si la case n'est pas révélée, alors on sort de la fonction.

On crée une liste d'une liste d'entiers qui va permettre de trouver toutes les cases adjacentes.
On fait une boucle qui prend une liste d'entiers et va parcourir les cases adjacentes,

S’il trouve une mine à la position 0 de la liste pour les lignes et à la position 1 de la liste pour les colonnes , alors on incrémente la valeur de mine adjacente de 1.

On met le nombre de mines adjacentes dans la variable minesAdjacentes de la fonction revelerCase (la case va donc afficher le nombre de mines adjacentes).

Si la case ne contient pas de mines adjacentes,
alors on fait une boucle qui parcourt la liste des cases adjacentes et on rappelle la fonction revelerCase avec en paramètre le premier élément à la position 0 et en deuxième paramètre le deuxième élément de la liste à la position 1 et qui recommence tout depuis le début (Peut ainsi provoquer une réaction en chaine)

Conclusion

Nicolas Petronis : J’ai pris grand plaisir à réaliser cette Saé en compagnie de Yuness. En effet j’ai largement préféré réaliser ce projet en Java que le Taquin en langage C.
Je trouve que le projet qu’on a réalisé est plutôt clair dans le nom des variables et des fonctions, c’est ce qui nous a beaucoup facilité la tâche quand il fallait créer de nouveaux fichiers et utiliser les fonctions de l’un ou de l’autre, j’en ai tiré beaucoup de satisfaction.

Yuness Soussi : Féru du langage Kotlin, j’ai longtemps attendu de pratiquer le Java au sein de l’IUT, voulant devenir développeur mobile android, cette Saé m’a permis de mieux cerner les divers aspects du Java. C’est aussi la première Saé que j’ai réalisée avec Nicolas Petronis, bien que nous nous parlions peu avant la Saé, une véritable synergie s’est créé et ça a été un véritable plaisir de créer le Démineur avec lui.

Yuness Soussi & Nicolas Petronis : Malgré quelques difficultés dans la réalisation de la Saé, nous sommes parvenus à réussir quasiment tous nos objectifs et ceci dans le délai accordé, nous avons tiré de cette Saé une certaine satisfaction mais surtout beaucoup d’expérience.
On avait ajouté le fait que, si on clique une fois sur une case vierge, alors il y a un point d’interrogation et si on clique une deuxième fois il y a un drapeau. Cependant on l’a enlevé car on a eu un problème avec la sauvegarde. On a donc préféré avoir une bonne sauvegarde et un jeu presque complet, plutôt que d’avoir une mauvaise sauvegarde avec la fonctionnalité de pouvoir mettre un point d’interrogation puis un drapeau. On reste malgré tout très contents d’avoir pu réaliser ce projet car malgré ce petit souci, on a réalisé la majorité de ce qui était demandé.

FIN