Waché Oriane
Gaffour Said-Waahid
Corret Bryan
1A2B

BUT 1 Informatique
IUT d'Orléans
Année 2021-2022
SAE 2.01

Développement d'une application
Le jeu du démineur
SAE 2.01

iUTO IUT ORLEANS INFORMATIQUE

UNIVERSITE D'ORLEANS

Sommaire:

Introduction:

Dans le cadre d'une situation d'apprentissage et d'évaluation du 2e semestre du BUT informatique qui cible la compétence 1 "Réaliser un développement d'application" (SAE2.01), nos enseignants nous ont demandé de développer en Java le jeu du démineur. Ce projet est à réaliser en groupe de 3 et avec des rendus échelonnés à faire afin que les enseignants puissent avoir un suivi de l'avancement de notre projet.

Sur ce projet nous devons donc développer toutes les bases du jeu comme le placement d'un drapeau, un compteur de drapeau, la victoire du joueur quand toutes les cases sont découvertes à l'exclusion des cases où se trouvent les bombes.

Afin de pouvoir réaliser ce projet nous devons passer par des phases de projets incontournables. Dans ces phases, on va trouver celle d'analyse, pour que l'ensemble du groupe ait acquis le sujet, cela passe par le travail de rédaction. Dans ce travail, nous allons écrire ce que nous avons compris, nos difficultés rencontrées, nos solutions apportées et la façon dont nous avons construit ce projet. On retrouve aussi la phase de développement, nous y retrouvons un travail de modélisation d'un diagramme de classes (Notre diagramme) afin de pouvoir mieux se projeter dans l'avenir, un dépôt git pour partager et avancer le codage à tous les membres du groupe, puis le codage complet et fonctionnel du projet pour le rendu final.

Nous devons réaliser un rapport qui consiste à mettre à l'écrit la phase d'analyse et la phase de développement en développant et détaillant à maxima tout le déroulement du projet.

Diagramme de classes:

Dans la première partie du travail, nous nous sommes posés pour réfléchir et analyser le projet, cela a permis de faire une première ébauche de notre travail. En passant par le cahier des charges, nous avons conçu une base de notre projet sur feuille. Nous devions produire un diagramme de classe, et par la suite un rapport contenant notre carnet de bord et des explications sur nos décisions prises lors du diagramme de classes. En se basant sur nos connaissances et nos cours, nous réalisons notre diagramme de classes sous forme de schéma qui représente les différentes relations et entités pour modéliser notre jeu afin de le transcrire en code java. On a réalisé notre diagramme de classes grâce à un site spécifique qui se nomme draw.io à la modélisation de toutes sortes de diagrammes.

Notre diagramme de classe:

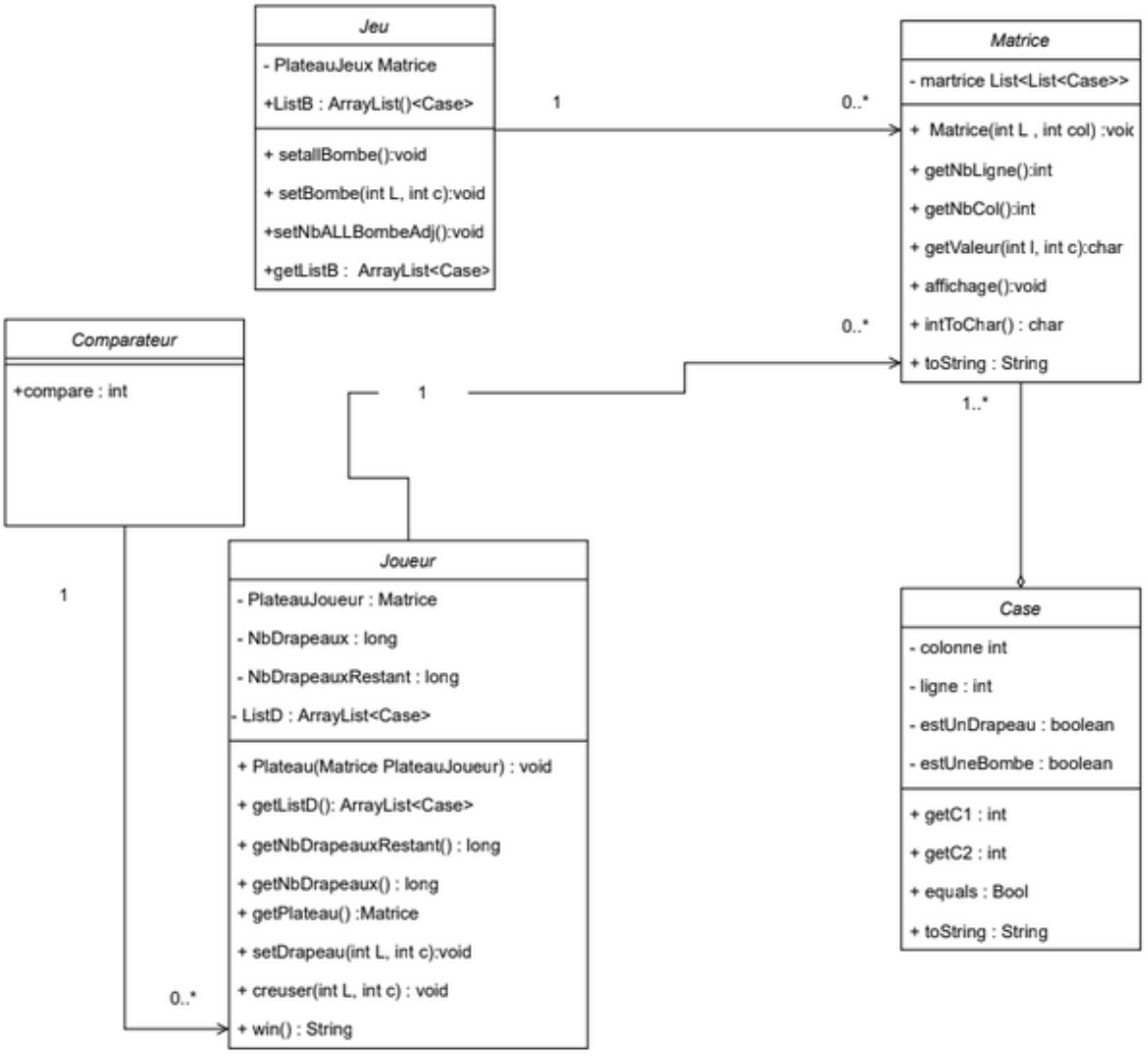
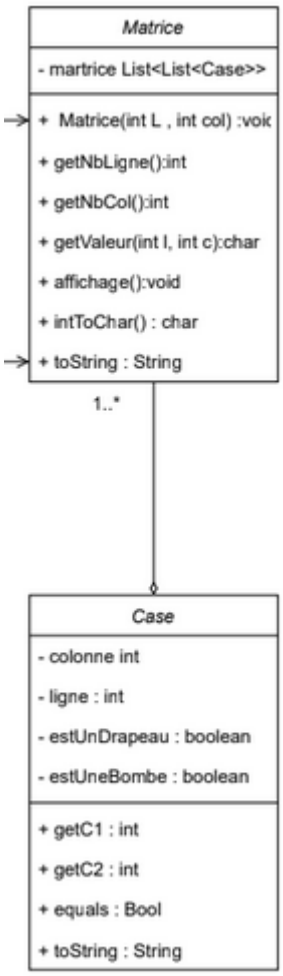


Diagramme de classe élaboré par le groupe



=> Nom de la classe
=> Attributs de la classe

=> Méthodes de la classe

=> 1..* = Multiplicité

=> Relation entre les deux classes

Extrait du diagramme de classe

Diagramme de classe fourni par les enseignants attitrés à la SAE:

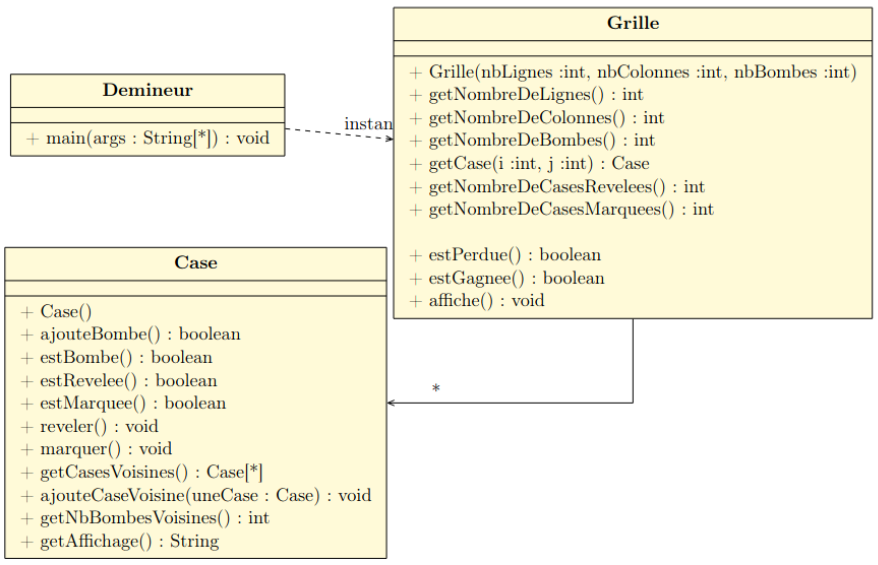


Diagramme de classe fourni par les enseignants attitrés à la SAE

Problèmes et solutions:

Par ailleurs, nous pouvons parler des problèmes rencontrés et des solutions apportées pour les surmonter. En effet, Nous avions du mal à choisir les classes que nous allions implémenter étant donné que certaines classes comme la classe Plateau et Matrice étaient similaires parce qu'elles réalisaient les mêmes fonctionnalités. Ainsi, nous nous sommes mis en accord de garder la classe Matrice puisque ce n'était pas nécessaire de garder la classe Plateau. Ensuite, nous avons rencontré des problèmes au niveau de la création du diagramme de classe étant donné qu'il y avait des différences entre le nôtre et celui des données par les enseignants.

Les différences:

Tout d'abord, les différences que nous avons notés entre les deux diagrammes de classes sont la présence de la méthode estPerdu() et estGagnee() qui sont présentent dans la classe Grille et renvoyant un booléen indiquant si le joueur a perdu la partie ou bien s'il a gagné, qui est présent dans la classe Grille du diagramme de classe fourni par les enseignants. En revanche, nous pouvons retrouver ces deux méthodes dans notre diagramme de classe sous la forme d'une seule méthode mais nommée différemment win() renvoyant cette fois çi une chaîne de caractère avec le renvoie d'un message déjà préalablement fixé. En effet, les renvoie de méthodes en booléen sont beaucoup plus faciles à manier et à utiliser étant donné que cela simplifie et aère le code pour assurer une bonne ergonomie au lieu d'avoir des lignes de code en plus pour que cela renvoie un résultat qu'on peut avoir facilement en le faisant différemment.

Dans un second temps, nous pouvons relever aussi une autre différence qui est présente entre les deux diagrammes qui sont notamment les méthodes getListB() dans la classe Jeu et getNombreDeBombes() que nous retrouvons dans la classe Grille du diagramme des enseignant. En effet, nous avons créé une liste de cases qui nous permet d'avoir une liste de bombes sur chaque ligne et colonnes de la grille du jeu du démineur ce qui permet de récupérer toutes les bombes.

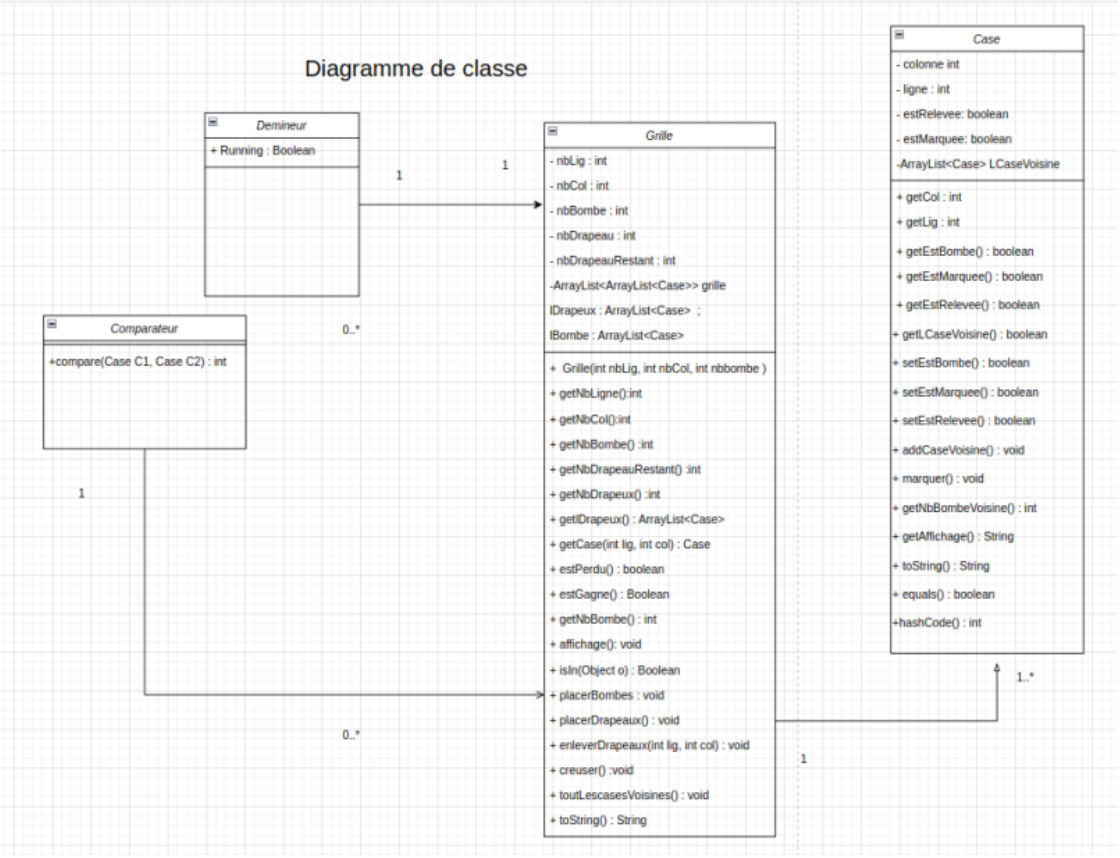
Enfin, au niveau des classes choisies dans notre première version du diagramme, nous avons remarqués que nous avons créer 2 classes à part (Matrice et Jeu) alors que dans le diagramme fourni, nous pouvons retrouver une classe qui s'appelle Grille qui englobe les attributs et méthodes présentent dans nos classes Matrice et Jeu, donc nous pouvons noter cette fusion qui est plus pratique car une classe gère toutes ces méthodes qui permettent de faire fonctionner notre jeu du démineur sur notre terminal de commande sous forme de menu que nous allons vous pourrez voir au niveau de l'exécution du code que nous vous fournissons.

Les similitudes:

Nous remarquons des similitudes dans ces deux diagrammes de classes. Pour commencer, dans notre classe Matrice qui correspondrait à la classe Grille du diagramme fourni. Nous y retrouvons notamment les méthodes getNombreDeLignes(), getNombreDeColonne() et affichage().
Pour terminer, dans la classe Case, nous avons créé les méthodes suivantes:

- estUnDrapeau() qui renvoie un boolean, il correspond à estMarquee() qui renvoie un boolean aussi.
- estUneBombe() qui renvoie un boolean, il correspond à estBombe() qui renvoie un boolean aussi.
- creuser(int L, int C), il correspond à estReveler(), ils ne retournent rien.
- toString(), il correspond à getAffichage(), qui renvoient une chaîne de caractère.

Résultat final du diagramme de classes:



Nouveau diagramme de classe élaboré par le groupe

Implémentation:

Dans la deuxième partie du travail, nous devions organiser nos fichiers dans différents dossiers qui sont: un dossier src pour tous les fichiers en java, un dossier bin pour tous les fichiers .class, et un dossier doc pour les fichiers de documentation. En respectant les bonnes pratiques comme le fait d'ajouter une documentation en commentant chacune de nos fonctions.

Dans le diagramme de classes, on remarque qu'il y a des attributs et des méthodes absentes. Pour commencer, sur la classe Case il manquait les variables suivantes :

```
private boolean estBombe = false;
  private boolean estRelevee = false;
  private boolean estMarquee = false;
  private int lig; // permet de savoir sur quelle ligne est la grille
  private int col; // permet de savoir sur quelle colonne est la grille
  private ArrayList<Case> LCaseVoisine ; // La liste des cases voisines
```

Extrait du code de la classe Case (L5-L10)

Nous avons des fonctions supplémentaires qui sont la redéfinition d'un equals et par conséquent du HashCode. La redéfinition du equals nous est utile pour comparer la liste des bombes et la liste des drapeaux. Cela permet de gagner du temps d'exécution car sinon nous devions charger tout mon plateau et regarder si toutes les bombes avaient la variable estRelevee à "true".

En contrepartie, une méthode nous a posé problème qui est "creuser" et plus précisément sa fonction récursive. En effet, lors de notre première version de "creuser" une case donnée, nous plaçons toutes les bombes avant l'appel cette méthode, ce qui faisait que l'on pouvait perdre dès le premier tour, ce qui est injuste pour le joueur.

```
public void creuser(int lig, int col) {
     this.getCase(lig, col).setEstRelevee(true);
     }
```

Extrait du code de la classe Grille (L220-237)

Concernant la classe Grille, comme pour la classe Case nous avons ajouté des méthodes qui sont toutLescaseVoisines() , isIn(), et des variables qui sont:

```
   private int nbLig;
   private int nbCol;
   private int nbbombe;
   private int nbDrapeux;
   private int nbDrapeuxRestant = 1;
   private ArrayList<ArrayList<Case>> grille;
   private ArrayList<Case> lDrapeux ;
   private ArrayList<Case> lBombe;
```

Extrait du code de la classe Grille (L6-13)

Dans notre deuxième tentative, nous avons réussi à "creuser" les cases voisines cependant nous découvrons que toutes les cases possèdent une bombe ce qui fait que le joueur perdait forcément la partie. Une fois ce problème résolu, nous n'arrivions pas à sortir

de notre fonction ce qui rendait le jeu impossible à jouer. La troisième tentative est la bonne, nous arrivons maintenant à creuser les cases voisines, éviter les bombes, placer des drapeaux, le problème était dû à une mauvaise condition pour exécuter la récursivité.

Lors de l'implémentation, nous avons rencontré un problème avec la grille visible dans le terminal, cette fonctionnalité ne faisait qu'aggraver le problème déjà existant, n'étant pas une chose vitale à l'exécution de ce programme nous avons préféré nous occuper d'autre problèmes comme ceux détailler au-dessus.

Dans le cadre de notre jeu, il y aura des fonctionnalités supplémentaires en plus de celles disponibles sur le jeu de base. Sur le projet, nous avons eu l'idée de pouvoir rendre le jeu accessible pour tout le monde, afin de répondre à cette problématique nous pourrons choisir la difficulté. La difficulté sera notamment calculée de deux manières soit par la taille du plateau et/ou par le nombre de bombes sur un plateau. Les paramètres par défaut sont réglés sur un plateau de 6x6 (6 lignes, 6 colonnes) avec 4 bombes placées.

Travail de groupe:

Lors de notre répartition des tâches à réaliser pour aboutir ce projet, nous avons mis en place un git qui nous permet de développer plusieurs fichiers en même. Nous utilisons des branches pour pouvoir faire cette dernière sans détruire le travail réalisé par un autre. Nous nous servons aussi de la fonctionnalité drive de google et de nos e-mail pour partager des fichiers de code quand nous pouvions pas utiliser le git. De plus, nous avançons ensemble sur un document drive pour produire le rapport de projet.

Bilan:

Pour conclure cette SAE, nous avons réussi à réaliser entièrement ce projet malgré nos difficultés pour plusieurs fonctions, cependant on a beaucoup sollicité l'aide des trois membres du groupe pour pouvoir continuer dans l'avancée du codage. Ainsi, grâce à cette avancée, nous avons réussi à implémenter notre IHM malgré les obstacles qu'on a pu rencontrer tout au long de ce projet ne serait-ce que dans la réalisation de notre diagramme de classes et la traduction de ce dernier en code java afin d'obtenir le résultat escompté qui est de développer le jeu du démineur.

Annexes

Journal de bord:

Date	Noms	Travail réalisé	Temps
27/04	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Prise de connaissance du sujet ● Commencement de la rédaction du rapport	30 min 30 min 30 min
2/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Choix des classes pour la création du diagramme de classes après la mise en commun avec l'équipe.	1h 1h 1h
3/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Choix définitif des classes après l'avis apporté par un professeur ● Ajout des attributs et méthodes de chaque classe à implémenter. ● Ajout des relations entre les classes	1h30 1h30 1h30
10/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Avancement du rapport de projet	1h30 1h30 1h30
17/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Avancement du rapport de projet	1h30 1h30 1h30
24/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Avancement du rapport de projet	1h30 1h30 1h30
31/05	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Avancement du rapport de projet	1h30 1h30 1h30
07/06	Waché Oriane, Gaffour Said-Waahid, Corret Bryan	● Préparation pour la soutenance	1h30 1h30 1h30