|  |  |
| --- | --- |
| 420-4C6  TP3 (25%)  Jeu de Démineur | |
| **Nom, prénom** | * Meyou Lemanga, Chris Tania * Tavares, Véronique |
| **Cours** | **420-4C6-LI Validation et maintenance des applications** |
| **Enseignant** | Samir Mendil |
| **Date exigée** | Voir Léa |
| **Évaluation** | Ce travail se fait en équipe de 2 |

|  |  |
| --- | --- |
|  | 1. En annexe vous trouveriez un exemple de code source pour le jeu de Démineur, actuellement le code permet l’affichage en mode Swing et le code ne respecte pas la norme MVC ni les principes SOLID et en plus aucune classe de test n’est fournie. Cela dit vous devez appliquer des modifications sur le projet fournie pour respecter les exigences suivantes :    1. Dans le menu Partie de l’interface du jeu, ajouter l’option d’affichage en mode JavaFX;    2. Les images dans le projet java sont volontairement corrompus, vous devez les remplacer par des nouvelles images, j’annexe quelques images que vous pouvez utiliser dans le dossier Ressources Images ;    3. Adapter le code pour appliquer le design pattern MVC;    4. Appliquer les principes SOLID;    5. Adapter le jeu pour appliquer le Design Pattern Factory lors du choix du mode d’affichage et de la difficulté du jeu (débutant, intermédiaire et expert);    6. Appliquer les tests unitaires (Plan test et un Tableau nombre tests /classes, durée, justification) du package Model en respectant la forme **Given**-**Then**-**When;**    7. Chaque équipe doit avoir un dépôt de sources « GIT » (historique des commit, GitHub ou Bitbucket) et en donner l’accès au professeur dès le début du développement. Mon courriel est samir.mendil@cegeplimoilou.ca |

**Règles du jeu.**

Le jeu du démineur est constitué d’une grille à deux dimensions, nommé le terrain, dans lequel se trouvent des mines. En début de partie, les mines sont invisibles et aucune parcelle de terrain n’est découverte. Le but est de déminer le terrain en marquant l’emplacement des mines et en découvrant le reste du terrain.

Pour ce faire, le joueur doit cliquer sur une parcelle de terrain. Celle-ci se découvre : il peut alors s’agir d’une mine ou d’un terrain vide. Dans le cas d’un terrain vide, tout le terrain vide adjacent se découvre aussi et des chiffres apparaissent sur quelques parcelles de terrain. Ces chiffres représentent le nombre de mines autour de la parcelle, partant de 1 à 8. Cependant, il n’est nullement indiqué dans quelles directions se trouvent ces dernières et le joueur doit donc se servir des autres parcelles de terrain pour le deviner.

Une fois que le joueur a identifié une mine, il doit la marquer. Si le joueur identifie correctement l’emplacement de toutes les mines, il gagne la partie. Par contre, s’il y a une parcelle de terrain qui n’est pas une mine, alors le jeu continue.

Pour compléter le jeu, il faut que le joueur tente sa chance sur plusieurs parcelles de terrain. Advenant le cas où que le joueur clique sur une parcelle de terrain contenant une mine, ce denier explose et perd la partie.

   

**Figure 1 – De gauche à droite, le début d’une partie, une partie commencée, une partie perdue et une partie gagnée**

Ici, le véritable but du jeu est de réussir à trouver les mines le plus rapidement possible. Il y a d’ailleurs plusieurs niveaux de difficulté, soit « Débutant », « Intermédiaire » et « Expert ». À chaque niveau, la taille du terrain augment, de même que le nombre de mines. Il est même possible de choisir exactement la taille du terrain ainsi que le nombre de mines.

**Conditions de réalisation du travail**

Ce travail, d’une valeur de 25% de la note finale, s’effectue en équipe de deux (2) ou de trois (3)

Chaque participant de l’équipe devra s’assurer d’effectuer une quantité de travail équitable. Il est bien entendu déconseillé de diviser les tâches directement : chaque membre de l’équipe devrait normalement travailler sur tous les aspects du projet afin de se préparer à l’examen.

**User stories**

|  |  |  |
| --- | --- | --- |
| No | Description | OK |
| 1 | En tant qu’utilisateur, je désire pouvoir commencer une nouvelle partie de « Démineur » dans le mode de difficulté sélectionné. Le bouton sourire doit lui aussi démarrer une nouvelle partie lorsque l’on clique sur ce dernier. |  |
| 2 | En tant qu’utilisateur, si je clique avec le bouton gauche de la souris sur une case non découverte du jeu, celle-ci doit se découvrir. Si la case possède des mines en guise de voisin, elle doit en afficher le nombre. Par contre, dans le cas contraire, elle doit aller découvrir les cases voisines. |  |
| 3 | En tant qu’utilisateur, je désire savoir le nombre de mines restant sur le champ de mines via un champ à gauche du bouton sourire. |  |
| 4 | En tant qu’utilisateur, je désire connaître le temps écoulé, en secondes, depuis la découverte de la première case, via un champ placé à droite du bouton sourire. |  |
| 6 | En tant qu’utilisateur, si je clique avec le bouton droit de la souris sur une case non découverte du jeu, celle-ci doit alterner entre les états suivants : Vide – Marquée d’un drapeau – Marquée d’un point d’interrogation. |  |
| 7 | En tant qu’utilisateur, lorsque je marque une case d’un drapeau, le nombre de mines doit décrémenter de 1. Il est bien entendu possible d’avoir plus de drapeaux que de mines sur la grille, ce que veut dire que le compteur peut très bien être négatif. |  |
| 8 | En tant qu’utilisateur, je dois être incapable de découvrir une case marquée. |  |
| 9 | En tant qu’utilisateur, je dois perdre la partie si je clique sur une case minée. |  |
| 10 | En tant qu’utilisateur, lorsque j’ai perdu, je désire savoir où se trouvaient les mines. |  |
| 11 | En tant qu’utilisateur, lorsque j’ai perdu, je désire savoir quels sont les emplacements marqués de drapeaux où il n’y avait pas de mine. |  |
| 12 | En tant qu’utilisateur, je dois gagner la partie s’il ne reste à découvrir que des cases avec des mines. |  |
| 13 | En tant qu’utilisateur, lorsque je gagne, je désire que les cases non marquées d’un drapeau où il y a une mine soient marquées d’un drapeau. |  |
| 15 | En tant qu’utilisateur, lorsque j’ai gagné ou perdu, je ne peux plus cliquer sur aucune case. Je n’ai pas d’autres choix que de commencer une nouvelle partie. |  |
| 16 | En tant qu’utilisateur, en cours de partie, je veux que le bouton sourire affiche l’état « Normal ». |  |
| 17 | En tant qu’utilisateur, lorsque je perds, je veux que le bouton sourire affiche l’état « Mort ». |  |
| 18 | En tant qu’utilisateur, lorsque je gagne, je veux que le bouton sourire affiche l’état « Lunette de soleil ». |  |
| 19 | En tant qu’utilisateur, je désire pouvoir choisir entre 3 niveaux de difficulté, à savoir « Débutant » avec une grille de 8x8 de 10 mines, « Intermédiaire » avec une grille de 16x16 de 40 mines et « Expert » avec une grille de 30x16 de 99 mines. |  |

**Exigences pédagogiques supplémentaires**

***Bouton interactif***

Le bouton interactif est le bouton « Bonhomme sourire » dans la barre d’outils. Ce dernier sourit si le joueur est en vie, tandis qu'il meurt quand le joueur meurt. Cliquer sur ce bouton démarre une nouvelle partie. Lorsque la partie est gagnée, il porte des lunettes soleil.

  

**Les différents états du bouton interactif**

***Cases du jeu***

Les cases du jeu adoptent deux états différents : découvert ou non découvert. En plus de ces deux états, une case est soit un numéro, de un (1) à huit (8), soit une mine. Ajoutez en plus de tout cela trois marquages possibles sur les cases non découvertes : non-marqué, drapeau ou point d’interrogation.

            

**Les différents états d'une case de jeu**

Lorsque le joueur perd, la mine sur lequel le joueur a explosé est marquée en rouge. De plus, les mines qui sont mal identifiées (un drapeau à la mauvaise place) sont marquées d’un « X » rouge.

 emptyFlat2

**D’autres états d’une case, à la fin du jeu où le joueur a perdu**

**Changements pour appliquer le principes SOLID**

|  |  |
| --- | --- |
| S : Chaque classe doit avoir une responsabilité unique | * Création de la classe DemineurJavaFX pour gérer le code en JavaFX dans une classe séparée et ne pas mélanger ce code avec le code Swing |
| O : Ouvert/Fermé | * D’autres classes peuvent implémenter les interfaces du menu et/ou extends MenuFactory, sans affecter son code ni celui des autres classes qui l’implémente * Extraction de la méthode selectionDeLaBonneDifficulte(int type) du constructeur et ajout à l’Interface |
| L : Héritage | * La classe MenuFactory et Interfaces du menu |
| I : Principe de ségrégation des interfaces | * Deux Interfaces ont été créées pour le menu, une pour la gestion du niveau de difficulté et l’autre pour le changement de mode.   + Cela rend le code plus réutilisable, si une sous classe veut simplement implémenter la gestion du niveau de difficulté et non le changement de mode par exemple. |
| D : Principe d'inversion des dépendances | * La classe enfant model/demineur peut être modifiée sans modifier le code de sa classe parent MenuFactory. |

**Plan test**

Toutes les méthodes des Beans du package modèle seront testées en respectant la forme Given-Then-When tels que demandé.

**Tableau du nombre de tests /classes, durée, justification**

**Justification pour toutes les classes :** Les Set et Get sont testés dans la même méthode de test, puisqu’il y a seulement des constructeurs par défaut dans les classes à tester et donc nous n’avons pas le choix de setter la valeur avant de la get.

|  |  |  |  |
| --- | --- | --- | --- |
| **Classe** | **Nombre des tests** | **Durée** | **Justification** |
| DeminCase | 6 | 61ms | La méthode test setEtGetGr prend plus de temps que les autres (22ms) pisqu’elle doit créer un objet Graphisme du package view. |
| Demineur | 7 | 38ms | La méthode test setEtGetJeux prend plus de temps que les autres (11ms) pisqu’elle doit créer un objet tableau d’objet DeminCase du package view. |
| MenuFactory | 1 | 30ms |  |
| Personaliser | 4 | 471ms | La méthode test setEtGetDemin prend beaucoup de temps (441 ms) puisqu’elle créer un objet Demineur avec paramètres du package view.  L’utilisation d’un mock aiderait à accélérer la vitesse du test. |
| Segment | 3 | 41ms | En plus des set et get, je test que, lors de l’appelle d’une case du tableau de boolean, celle-ci retourne la bonne valeur. |
| Temps |  |  |  |

**Plan de test de la user story 17 : En tant qu’utilisateur, lorsque je perds, je veux que le bouton sourire affiche l’état « Mort ».**

GIVEN : Créer les objets qui seront utiles aux tests

* Demineur

WHEN : setter les attributs appropriés qui définissent que l’utilisateur à perdu.

* (demineur.getJeux()[y][x].getDeminCase().getEtat() == 0 || demineur.getJeux()[y][x].getDeminCase().getEtat() == 3)
* && demineur.getJeux()[y][x].getDeminCase().isMine()

THEN : Vérifier que l’Icon du bouton sourire est bien à l’état « Mort » (oups).

**Nombre d’assertions à effectuer** : une (1) qui vérifie que le l’Icon du bouton sourire est bien celle du « Mort » (oups), en plus des Get (3) et Set (2) des attributs utiles à faire en sorte que l’utilisateur à perdu.

**Durée de codage** : environ 25 minutes

**Durée d’exécution** : environ 700ms.

**Importance du test** : Importance faible car le jeu peut fonctionner sans que le bouton sourire s’affiche à l’état mort lorsque l’utilisateur à perdu. C’est seulement pour rendre le jeu plus esthétique et elle n’impacte aucune autre fonctionnalité si elle n’est pas fonctionnelle.