Checklist itération 2 (non-officielle)

> **Attention** cette checklist n'est pas officielle, elle est seulement basée sur le PDF de l'itération 2 et général. Pour la checklist officielle cliquez ici

Contraintes générales

☑ La structure des fichiers corresponds à celle montrée dans le PDF

☑ Le projet utilise la version 11 de Java et intègre l'interface graphique Swing

☑ Le projet est compatible avec la version d'Eclipse de référence (voir ressources informatiques sur Learn)

☑ Les tests unitaires sont effectués à l'aide de JUnit5

☑ Le projet utilise le plug-in PMD avec le ruleset donné sur Learn

Acquis d'apprentissages

☑ Respect des principes de la programmation orientée objet

☑ Les comportements du programme sont testé à l'aide de tests unitaires

☑ Le programme est bien documenté à l'aide de la JavaDoc (et calcul de la CTT si demandée)

☑ Les utilisations des interfaces de collections et de ses implémentations sont justifiées

Réponse feedback

☑ Faire une copie défensive de `getCoordinates()` (par exemple à l'aide de `Collections.unmodifiableSet(Set)`)

☑ Passer la case active dans Game plus tot que Player

☑ Potentiellement faire passer les messages de statistiques sur le jeu dans Game plus tot que superviseur

☑ Rendre toutes les classes `final` pour empécher d'en hériter

☑ Supprimer toutes les définitions de `equals` et `hashcode` quand ce n'est pas nécessaire (par exemple Player)

☑ Faire en sorte que MainMenuSuperviser crée le jeu et que PlayGame le récupère

☑ Rendre la classe Coordinate plus intelligente pour Player.move(deltaX, deltaY)

☑ Expliquer le `+` dans le calcul de la CTT du placement des trésors dans CaseMap

☑ Faire en sorte que la factory crée la map et l'injecte dans le jeu

Fonctionalités

AI-4 : Changer de case active

En tant que Joueur, je souhaite changer de case active pour poursuivre mon exploration.

☑ Appuyer une touche fléchée déplace la case active.

☑ Changer de case active met à jour le type et le cout de la case active affichée.

☑ A priori, le joueur peut se déplacer hors des limites de la carte. On peut voir une case hors des limites d'une carte comme une case de type Eau

Tests d'acceptation

Les tests d'acceptation partent de la Carte 1. On suppose que la case active de départ est celle du centre (encadrée en rouge). Attention, nous pouvons jouer les tests d'acceptation sur d'autres cartes.

☑ Etant donnée une partie démarrée avec la carte donnée en exemple, quand j'appuie sur la flèche du haut, alors le sélecteur se déplace sur la case (0,1) de type Sable et de cout 1.

☑ Etant donnée une partie démarrée avec la carte donnée en exemple et la case active placée sur la case (0,1), quand j'appuie sur la flèche du bas puis sur la flèche de droite, alors le sélecteur est sur la case (1,2) de type Prairie et de cout 2.

☑ Etant donnée une partie démarrée avec la carte donnée en exemple et la case active placée sur la case (1,2), quand j'appuie sur la flèche de bas, alors le sélecteur se déplace sur

la case (2, 2) de type Eau et de cout 0.

AI-5 : Creuser une case

En tant que joueur, je souhaite creuser la case active afin de trouver des trésors

☑ Si cette case est de type Eau, rien ne se passe.

☑ Si cette case est de type autre que Eau et sans trésor, l'application affiche le sprite « Dug ».

☑ Si cette case possède un trésor, l'application affiche le sprite « Treasure ».

☑ Si cette case ne possède pas de trésor, la bourse du joueur est diminuée du cout de la case.

☑ Si cette case possède un trésor, la bourse du joueur est diminuée du cout de la case et augmentée de la valeur du trésor découvert.

☑ Si cette case possède un trésor, le nombre de trésor à trouver est diminué de 1.

☑ Si le joueur tente de creuser alors qu'il est en dehors de la carte, rien ne se passe : en particulier il n'y a pas de messages d'erreur remontés par l'application.

Tests d'acceptation

Les tests d'acceptation partent de la Carte 1. On suppose que le seul trésor se trouve en case (1,2). Attention, nous pouvons jouer les tests d'acceptation sur d'autres cartes.

☑ Soit une partie démarrée dont la case active est (0,0), quand j'essaie de la creuser, rien ne se passe.

☑ Soit une partie démarrée dont la case active est (0,1), quand je la creuse, alors l'application retire 1 pièce de ma bourse.

☑ Soit une partie démarrée dont la case active est (0,1) et est déjà creusée, quand je la creuse, rien ne se passe.

☑ Soit une partie démarrée dont la case active est (1,2), quand je la creuse, alors ma bourse perd 2 pièces et reçoit autant de pièces qu'en compte le trésor.

☑ Soit une partie démarrée dont la case active est (1, 0), quand je la creuse, alors rien se passe.

AI-6 : Fournir des indices

En tant que Joueur, quand je creuse une case, je souhaite avoir un indice sur le trésor le plus proche afin de trouver des trésors

Une case peut contenir un indice sur le trésor le plus proche prenant la forme d'une flèche.

☑ La flèche d’une case indique la direction du trésor le plus proche.

☑ Un trésor peut être le plus proche de toutes les cases situées dans un carré de 5 cases de côté et centré sur le trésor.

☑ Il y a 8 directions possibles : O, NO, N, NE, E, SE, S, SO.

☑ Seules les cases creusables sans trésor peuvent contenir un indice.

☑ Quand une case C se trouve à égale distance de deux trésors, l’application choisit celui de plus grande valeur. Quand une case C se trouve à égale distance de deux trésors de même valeur, on privilégie celui dont la position est la plus proche du coin supérieur gauche.

☑ La distance entre deux cases c1 = (row1 ; col1) et c2 = (row2 ; col2) vaut $\sqrt{(row1 - row2)^2 + (col1-col2)^2}$

☑ Les indices ne sont pas modifiés en cours de partie, ce qui implique qu’ils peuvent être tous calculés au début de la partie.

Tests d’acceptation

Nous partons de la Carte 2 pour décrire les tests d’acceptation. Les trésors y sont prédisposés avec les valeurs entre parenthèses. Attention, nous pouvons jouer les tests d’acceptation sur d’autres cartes.

☑ Étant donnée une partie démarrée avec la Carte 2, quand je creuse la case (4, 4), je découvre une flèche orientée vers le bas.

☑ Étant donnée une partie démarrée avec la Carte 2, quand je creuse la case (3, 5), je découvre une flèche orientée vers le Sud-Ouest.

☑ Étant donnée une partie démarrée avec la Carte 2, quand je creuse la case (3, 1), je découvre une flèche orientée vers le Nord-Ouest.

☑ Étant donnée une partie démarrée avec la Carte 2, quand je creuse la case (8, 0), je ne découvre ni indice, ni trésor.

Phase de conception et problèmes à résoudre

Diagramme de conception générale

☑ Les Candidats et les Responsabilités ont été identifiées

☑ Les cartes CRC et les liens entre elles ont été faites

☑ Le diagramme de séquence/collaboration a été créé

☑ Le diagramme de classe a été fait

Controller le placement des trésors pour les tests

☑ Déclarer une interface pour remplacer `Math.random`, `Random` et/ou `Collections.shuffle()`

☑ Implémenter cette interface en faisant appel à la/les méthodes ci-dessus

☑ Remplacer les dépendances à la méthode de base dans le code par l'interface

☑ Injecter la dépendence (ajouter l'objet implémentant l'interface dans le constructeur des classes qui ont besoin d'aléatoire)

☑ Implémenter cette interface pour les tests (elle peut par exemple retourner une collection de coordonnées fournies lors de la construction de l'objet)

☑ Utiliser cette implémnetation pour les tests

Gestion des indices

> **TODO**

Questions supplémentaires d'algo et de POO

Questions algorithmiques

☑ Ecrire les post-conditions qui seront vérifiées par le montant ajouté à la bourse du joueur (gain minimum d'un creusage et gain maximum d'un creusage) dans la javadoc de `PlayGameSuperviser.onDig()`

☑ Indiquer la CTT du calcul des indices à affecter aux cases creusables dans la javadoc en entête de TreasureQuestGame (Pour répondre à cette question, examinez la partie de votre code concernée, et identifiez les boucles, les imbrications, mais aussi les collections utilisées et leurs opérations ou les appels de sous-méthodes ou de méthodes d'autres objets. Quand vous répondez, n'oubliez pas d'indiquer à quoi correspondent vos libellés N,M, etc)

☑ Indiquer dans l'entête de la classe `TreasureQuestGame` quelle interface de collection a été utilisée pour représenter les coordonnées environantes (Justifiez votre choix en identifiant les principales opérations dont vous aurez besoin au cours de cette itération ? Avez-vous notamment besoin d'accéder à un élément précis ? Si oui, sur base de quelle sorte de clé ?)

☑ Indiquer dans l'entête de la classe `TreasureQuestGame` quelle implémentation de collection a été utilisée pour représenter ces coordonnées (justifier votre choix en déterminant les CTT des principales opérations pour l'itération 2)

Questions POO

☑ Les classes sont dans `treasurequest.domains` et n'ont pas de lien avec les vues (tout ce qui pourrait être en dehors de `domains`)

☑ L'encapsulation (attributs private) et les copies défencives sont bien effectuées sur toutes les classes du domains

☑ Ecrire des méthodes courtes et peu complexes (et tenter de respecter la règle de Demeter)

☑ Ecrire des classes timides (c'est à dire qui font le boulot et qui ne nécessite pas de tout le temps leur demander des choses depuis d'autres classes)

☑ Bon usage du polymorphisme

☑ Bonne utilisation des interfaces

☑ Aucune alerte PMD

☑ Tests unitaires JUnit5 dans le dossier `tests` qui donne un coverage d'au moins 90% sur chaque classe du domains (pour un degré de maitrise supplémentaire → coverage de 100% de ces classes)

☐ (degré de maitrise supplémentaire) Un plan de test exhaustif valide le code correspondant à la CTT demandée du placement des trésors et du calcul des indices

---

Revision #22
Created 2 May 2023 07:50:33 by SnowCode
Updated 21 September 2023 19:40:34 by SnowCode