Checklist itération 3 (non-officielle)

> **Attention** cette checklist n'est pas officielle, elle est seulement basée sur le PDF de l'itération 3 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

☑ Ajouter la `big-map.txt` dans le projet

☑ Télécharge l'archive `ai2023.results.zip` et décompresse son contenu dans le dossier `resources/images/results`

☑ Dans l'enum `treasurequest.supervisors.views.ResultType`, renomme la valeur `BALANCE` en `DURATION`

☑ Édite ensuite la constante `RESULT_TYPE` de la classe `treasurequest.swing.Theme` pour y ajouter les entrées correspondant aux statistiques comme ci-dessous.

```
/**
* Définit les images associées aux différents résultats.
* */
public static final Map<ResultType, Image> RESULTS_SPRITES = Map.of(
ResultType.NONE, new ImageIcon("resources/images/results/none.png").getImage(),
ResultType.LOSS, new ImageIcon("resources/images/results/loss.png").getImage(),
ResultType.GAIN, new ImageIcon("resources/images/results/gain.png").getImage(),
ResultType.DURATION, new ImageIcon("resources/images/results/duration.png").getImage(),
```

```
ResultType.TOURIST, new ImageIcon("resources/images/results/tourist.png").getImage(),
ResultType.FARMER, new ImageIcon("resources/images/results/farmer.png").getImage(),
ResultType.LUMBERJACK, new ImageIcon("resources/images/results/lumberjack.png").getImage(),
ResultType.MINER, new ImageIcon("resources/images/results/miner.png").getImage()
);
```

☑ Un bug empêche l’affichage de la bonne image. Pour le résoudre, modifie la ligne 79 de la classe `treasurequest.swing.GameOverSwingView` comme ci-dessous.

```
panels.add(new GameResultPanel(type, message, getWidth()/100*3 + col*(256+5), getHeight()/3 +
row*(200+5)));
```

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

☑ Coriger la valeur de M dans le calcul de la CTT

☑ Rendre Case en tant que non-data class

☑ Vérifier les références de Coord dans la méthode de fabrique de HintOrientation

☑ Complèter les préconditions et postconditions de toutes les méthodes publiques

☐ Changer les conditions de game over pour les cases creusable et leur coût

Fonctionalités

AI-7 Finir une partie

En tant que joueur, je souhaite savoir quand la partie se termine, afin de connaître mes résultats.

☑ La partie se termine si le joueur a découvert tous les trésors, s'il fait banqueroute ou s'il décide d'abandonner (voir itération 1).

☑ Le joueur fait banqueroute quand il n'a plus assez de pièces pour creuser les cases restantes. Autrement dit, la partie se termine quand il n'existe plus de cases pouvant être creusées par le joueur.

☑ Quand la partie est terminée, l'application redirige le joueur vers l'écran de fin de partie. Cependant, en cas d'abandon, l'application redirige le joueur vers le menu principal

Tests d'intégration

Les tests d'acceptation partent de la Carte 1 (voir ci dessous). On suppose également que le seul trésor à trouver est sur la case P, de type « Prairie ». Attention, nous pouvons jouer les tests d'acceptation sur d'autres cartes.

```
E S E
F R P
E S E
```

☑ Étant donné une partie débutée avec la Carte 1, quand je creuse la case de coordonnées (1,2), alors l'application affiche l'écran de fin de partie.

☑ Étant donné une partie débutée avec la Carte 1, quand je creuse les cases de coordonnées (0,1) et (2, 1), alors l'application affiche l'écran de fin de partie.

☑ Étant donné une partie débutée avec la Carte 1, quand j'appuie sur la touche « Esc », alors l'application affiche le menu principal

AI-8 Afficher les statistiques simples

En tant que joueur, je souhaite connaître les statistiques sur mes parties, afin de savoir si j'ai gagné ou perdu de l'argent.

☑ On souhaite connaître les dépenses que le joueur a faites. Ces dépenses correspondent à la somme des pièces dépensées par le joueur pour creuser les cases.

☑ On souhaite connaître les gains que le joueur a faits. Les gains correspondent à la somme des pièces que le joueur a reçues pendant ses parties.

☑ On souhaite connaître le temps de jeu du joueur. Le temps de jeu sera exprimé au format MM:SS, comme vous avez déjà eu l'occasion de le faire pendant les cours théoriques de POO.

☑ La durée d'une partie abandonnée reste comptabilisée dans le temps de jeu.

☑ On souhaite connaître le profil du joueur. Le profil correspond au type de la plus grande zone de cases adjacentes de même type que le joueur a creusées pendant une partie. Les profils sont sable → touriste, prairie → fermier, forêt → bucheron, rocher → mineur.

☑ Quand deux zones candidates sont de même taille, l’application privilégie la plus ancienne, c’est-à-dire celle des deux qui a été commencée en premier.

☑ Deux cases sont contigües si leur distance est inférieure à 2 afin de tenir compte des cases diagonales.

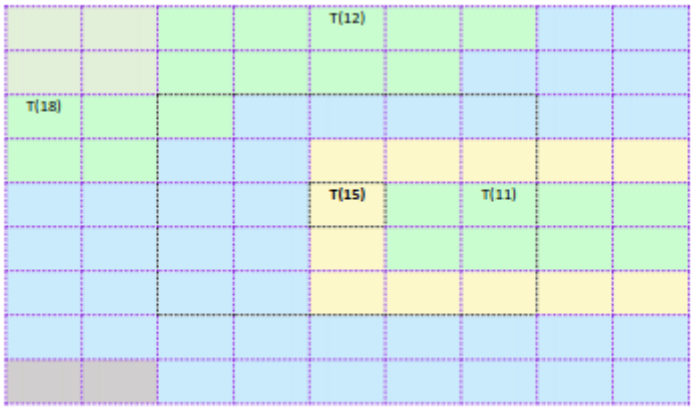
☑ Un document annexe présente un algorithme pour trouver la plus grande zone creusée.

☑ Ces statistiques perdurent d’une partie à l’autre.

☑ Les gains et les pertes en cas d’abandon restent comptabilisés. Cependant, le profil ne sera pas adapté.

Tests d'acceptation

Nous partons de la Carte 2 pour décrire les tests d’acceptation. On fait l’hypothèse que les trésors sont cachés dans les cases indiquées. Attention, nous pouvons jouer les tests d’acceptation sur d’autres cartes.



Carte 2 carte composée de plusieurs trésors

☑ Soit une nouvelle partie débutée avec la Carte 2 où le joueur a reçu 8 pièces, si je creuse les cases de (ligne : 5, colonne : 4), (5, 5), (5,6), (5,7) et (6,7) alors la partie se termine, j’ai dépensé 8 pièces, j’ai gagné 8 pièces (celles reçues en début de partie) et je suis fermier.

☑ Soit une nouvelle partie débutée avec la Carte 2 où le joueur a reçu 8 pièces, si je creuse les cases (ligne : 3, colonne : 4), (4, 4), (5, 4), (5, 5), (5,6), (4,6), (2,0) et (0,4), alors la partie se termine, j’ai dépensé 13 pièces, j’ai gagné 64 pièces (56 pièces + 8 reçues en début de partie) et je suis touriste.

☑ Soit une seconde partie débutée après la partie précédente, alors la bourse du joueur vaut 59 pièces (51 pièces encore disponibles et 8 pièces reçues en début de partie).

☑ Soit une nouvelle partie débutée avec la Carte 2, quand la partie se termine, le temps de jeu affiché par l’application respecte le format demandé.

☑ Soit une seconde partie débutée avec la Carte 2, quand la partie se termine, alors le temps de jeu affiché est plus grand que le temps de jeu affiché précédemment

AI-9 Jouer une partie sur une carte pseudo-aléatoire

☑ Un joueur peut lancer une partie qui se déroule sur un terrain aléatoire à l'aide d'un nouvel item « Partie aléatoire » affiché par le menu principal

☑ Le terrain correspond à une sous-zone carrée de 16 cases de côté extraite du fichier `resources/maps/big-map.txt` (voir les préalables).

☑ Cette sous-zone sera tirée aléatoirement

Tests d'acceptation

Quand je démarre une partie aléatoire, alors l'application m'affiche une carte carrée de 16 cases de côté.

☑ Quand je démarre une nouvelle partie classique, que je l'abandonne et que je démarre une partie aléatoire, alors l'application m'affiche une carte carrée de 16 cases de côté.

☑ Quand je démarre une partie aléatoire, que je l'abandonne et que je démarre une seconde partie aléatoire, alors l'application m'affiche une carte carrée de 16 cases de côté différente de la précédente

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

Questions supplémentaires d'algo et de POO

Questions algorithmiques

☑ Ecrire les conditions pour qu'une partie se termine dans la Javadoc de GameOverSupervisor (Détaillez-les précisément en fonction des états des objets manipulés par votre implémentation) → expliquer quels sont les objets qui détienne l'information

☑ Calculer et justifier la CTT du calcul de la plus grande zone de case adjacente dans la JavaDoc de la méthode qui effectue cette opération puis indiquer clairement dans GameOverSupervisor où cette méthode est implémentée

☑ Indiquer le choix de l'interface de collection utilisée pour mémoriser les cases visitées dans la JavaDoc de la classe où la CTT est calculée (voir précédemment)

☑ Indiquer le choix d'implémentation de l'interface de collection et justifier son choix en calculant la CTT des différentes opérations faites avec la collection et justifier. Indiquer le raisonnement dans la JavaDoc de la même classe (voir précédemment)

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

Revision #25
Created 9 May 2023 07:20:27 by SnowCode
Updated 21 September 2023 19:40:34 by SnowCode