Nom : SAADA KHELKHAL
Prénom : Sem
Groupe : 1

Nom : LABRAOUI
Prénom : Mohamed Wassim
Groupe : 1

Fiche de gestion de la maintenance du Projet Jeu de Mines.

Requête : Demande d'analyse dans le cadre d'une maintenance préventive

Prérequis :

Profil du mainteneur :

- Compétences pluridisciplinaires
- Savoir developper,concevoir,tester...
- Experience de travail avec les outils de maintenance
- Connaitre les meilleure méthode de maintenance
- avoir de l'experience avec le type de projet a maintenir
- Forte compétence en communication écrite et verbale pour interagir avec les utilisateurs et les clients.
- Capacité à s'adapter rapidement aux changements dans les exigences et les technologies.
- Connaissance des bonnes pratiques de programmation, telles que la documentation et la gestion de la qualité du code.

Pré-analyse :

1. **Compréhension et familiarisation avec le projet:**
   - Comprendre les spécifications
   - Examiner la documentation (si disponible)
   - Comprendre l'infrastructure du logiciel

- Comprendre les processus métier
- Comprendre les dépendances entre les processus metier
- Tester manuellement le logiciel
- Executer les test (si disponible)
- Reproduire les erreurs

Résultats Analyse structurelle :

SonarQube :

- Coverage : 0 % Cela indique que le code n'a pas été couvert par des tests
- Reliability : D Cela indique que le code a un certain nombre de problèmes de fiabilité
- Security : A Cela indique que le code est sécurisé
- Security Review : E Cela indique que le code n'a pas été examiné de manière exhaustive pour détecter des vulnérabilités de sécurité potentielles.
- Maintainability : A Cela indique que le code est facile à maintenir et à comprendre
- Duplication : 0 % Cela indique qu'il n'y a pas de code en double dans le projet.
- Issues : 59 Cela indique qu'il y a 59 problèmes détectés dans le code

CodeScene :

- Code Health average : 7.05 / 10
- Hotspot Code health : 7 / 10
- Code Health worst performance : 6.77 / 10
- Knowledge Loss, Average (Max) : 0 (0) % Abandoned Code
- Interactive hotspot map : Ces couleurs représentent différents niveaux de maintenabilité et de qualité, basés sur la note de santé du code. Le code rouge a une note faible et nécessite une attention immédiate, le code jaune a une note modérée et pourrait bénéficier de quelques améliorations, tandis que le code vert a une note élevée et est généralement en bon état.

    - Red code : 0 %
    - Yellow code : 91.3 %
    - Green code : 8.7 %
- Hotspots : 100 %
- Worst performer : 6.7 / 10

DeepSource :

- performance issues : 0
- Style issues : 0
- Documentation issues : 67
- Anti-patterns : 12
- Security issues : 2
- Coverage issues : 0
- Type check issues : 0
- Bug risks : 12

Cela suggère que Dans l'ensemble, le code nécessite des améliorations dans plusieurs domaines, mais il n'est pas dans un état critique.

Size :
- Lines of code : 303
- Lines : 383
- Statements : 182
- Functions : 7
- Classes : 3
- Files : 3
- Comment Lines : 1, 0.3 %

Complexity :
- cyclomatic complexity : 83
- cognitive complexity : 211

Synthèse Analyse Structurelle :
Les résultats de l’application des métriques en utilisant les différents outils SonarQube,.., reflètent :

- Une basse fiabilité du logiciel.
- Une securité assez bonne mais avec des potentiel probléms.
- Trés peut de duplication de code.
- Un nombre élever de probléme proportionelement a la taille du logiciel.
- Une complexité éleve du logiciel.

Résultats Inspection Manuelle :

Ressources évaluées	**Avis**
Code source	Toute la logique du logiciel et regrouper d'un une seul class java
Documentation	Indisponible ce qui rend la compréhension de l'intension du développeur difficile.
Tests unitaire	Indisponible ce qui rend la verification du respect des specification difficile
Structure du projet	Constitué de de classes seulement une pour la fonction main et une pour la totalité du jeu
Commentaire	Code non commenté

Synthèse de l'analyse personnelle :
Après l'inspection du code source de ce projet, nous constatons :

- L'absence de documentation.
- Aucun test unitaire.
- Code non commenté
- Respect les regles du jeu de mines

- Mauvaise utilisation de l’orienté objet (toute la logic du jeu en une seul class, mauvaise encapsulation, utilisation de numero magic sans variable).
- Le redemarrage d’une nouvelle partie prends en compte le clique ce qui rend le redemarge d’une partie avec un Board complet impossible.
- Beaucoup de fonction très complex (nested ifs, utilisation de while la ou for devrait étre utiliser, utilisation de if la ou un switch devrait étre utiliser …).

Post-Analyse :
Interprétation des Résultats (des deux analyses) et synthèse :
Après avoir analysé automatiquement et manuellement le projet nous déduisons que :

- La qualité globale du code est faible, avec des problèmes de fiabilité, de complexité et de maintenabilité.
- Le code ne respecte pas les bonnes pratiques de programmation orientée objet.
- Il n'y a pas de documentation ou de tests unitaires pour aider à la compréhension et à la vérification du code.
- Il y a beaucoup de code complexe et difficile à suivre, avec de nombreuses boucles imbriquées et des conditions complexes.

Décision
Suite aux conclusions de l'analyse, nous recommandons d'effectuer une refonte complète du code en utilisant des bonnes pratiques de programmation orientée objet et en divisant le code en classes distinctes pour faciliter la compréhension et la maintenance. Nous recommandons également de mettre en place une documentation

appropriée et des tests unitaires pour assurer la qualité et la fiabilité du code. Il est également recommandé de corriger la fonctionnalité de redémarrage du jeu pour permettre une réinitialisation correcte du jeu. Enfin, nous suggérons de réduire la complexité du code en utilisant des boucles et des conditions plus simples lorsque cela est possible.