Tobias Lepoutre (20177637)

Abderrahmane Manseur (20148685)

Devoir 2 – IFT3913

Métrique 1: **Taux de couverture de code par les tests**.

- Raisonnement: Un taux élevé de couverture indique que la majeure partie du code est testée.
- Comment mesurer: Utiliser des outils comme JaCoCo ou Cobertura.

Métrique 2: **Nombre de tests par classe/module**.

- Raisonnement: Indique la densité des tests pour chaque unité de code.
- Comment mesurer: Examiner le code des tests.

Métrique 3: **Date du dernier commit pour les tests vs. Date du dernier commit pour le code**.

- Raisonnement: Analyser la fraîcheur des tests par rapport au code, afin de vérifier si les tests sont à jour.
- Comment mesurer: Analyser l'historique des commit github et ensuite compter le nombre de fichiers de test qui n'ont pas reçu de commit après que leur code source ai reçu un commit et diviser par le nombre total de fichier de test.

Métrique 4: **Complexité cyclomatique moyenne des tests**.

- Raisonnement: Mesure directement la complexité des tests.
- Comment mesurer: Parcourir tout les fichier de test et calculer leur complexité cyclomatique moyenne ensuite faire la moyenne des moyennes de tous les fichiers de test.

Métrique 5: **Taux de méthode de test contenant un javaDoc (Maintenance)**

- Raisonnement: Si les méthodes de test contiennent des javaDocs, ceci améliore grandement la maintenance des fichiers de test car ça permet de comprendre facilement leur fonctionnement.
- Comment mesurer: Parcourir les méthode de test afin de compter combien d'entre eux contiennent du javaDocs, ensuite diviser par le nombre total de méthode de test afin d'avoir un taux.

Métrique 6: **Taux de méthode de test non void qui contiennent une documentation @return**

- Raisonnement: Si les méthodes de test qui retourne une valeur contiennent une documentation sur la valeur de retour, ceci aide pour la maintenance afin de savoir quelle est la nature de cette valeur retourner.
- Comment mesurer: Parcourir les méthode non void dans les fichiers de test afin de vérifier lesquels contiennent un @return, les compte et diviser par le nombre total de méthode de test non void.

Choix des métrique pour chaque question:

Q1: Est-ce qu'il y a assez de tests?

Métrique1 et Métrique 2.

Q2: Est-ce que les tests sont à jour avec le reste du code?

Métrique 1 et Métrique 3.

Q3: Est-ce que les tests sont trop complexes?

Métrique 2 et Métrique 4.

Q4: Est-ce que les tests sont suffisamment documentés?

Métrique 5 et Métrique 6.

Données:

Métrique 1 : Cette métrique nous donne une valeur de 25,43%.

Métrique 2 : Cette métrique nous donne une valeur de …

Métrique 3 : Cette métrique nous donne une valeur de 0.

Métrique 4 : Cette métrique nous donne une valeur de 0,15.

Métrique 5 : Cette métrique nous donne une valeur de 94,98%.

Métrique 6 : Cette métrique nous donne une valeur de 79,49%.

Question 1:

Il y a presque assez de test étant donné que nous avons une assez bonne couverture.

Question 2:

Après l'analyse des données des métrique proposé, on remarque déjà en observant la métrique 3, le nombre de tests qui n'ont pas été mis à jours, on observe un ratio de 0, ce qui veut dire que tous les fichiers de test ont reçu une mise à jours de leur code après modification des méthode qu'ils testent, ce qui signifie que les tests sont bien à jours avec le reste du code, malgré que la couverture n'est pas très élevée.

Question 3:

On remarque après avoir évaluer la complexité cyclomatique que les fichier de test sont très peu complexe, car ils ont une moyenne de complexité de 0,15, ce qui est trés trés faible, ceci signifie que leur fonctionnement est relativement simple, c'est à dire qu'il ont très peu de branche de décision effectuer, leur maintenance sera relativement simple car ils sont facile à comprendre.

Question 4:

Avec les donnés collecté on remarque que les méthode de test sont très bien documenté étant donné que nous avons trouvé que 94,98% des méthode de test contiennent des javaDocs et la plupart des méthode non void contiennent même de la documentation sur leur valeur de retour (presque 80%) le reste des méthode qui en contiennent pas ont probablement un retour trivial, ce qui signifie que les tests seront très facile à maintenir car on comprend facilement leur utilité leur usage etc..