

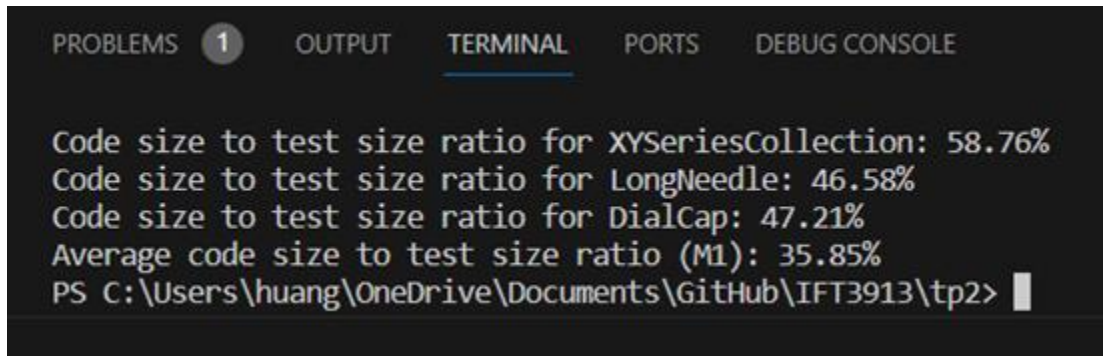
Rapport TP2 du cours IFT3913

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

- Métrique M1 : Ratio taille du code / taille des tests
- Métrique M2 : TPC (Nombre de tests par classe)

Explication : La métrique M1 donne une idée générale du niveau de test par rapport à la taille du code. Un ratio élevé pourrait indiquer un manque de tests. M2 se penche sur combien il y a de tests pour chaque classe, ce qui permet d'identifier si certaines classes sont sous-testées.

Résultats : À partir des résultats obtenus avec les scripts 'tloc' et 'TestCoverageMetrics', nous pouvons affirmer que la couverture de tests pour 'jfreechart' est satisfaisante. Le ratio obtenu de la taille du code sur la taille des tests est de 35.85%. Dans notre analyse, nous considérons qu'un programme est suffisamment testé si ce ratio est inférieur à 50%. Un ratio de 50% signifie que la taille des tests représente la moitié de la taille du code, ce qui est déjà significatif. En d'autres termes, un ratio de 50% implique que pour chaque ligne de code, il y a 0.5 lignes de tests. Dans notre cas, un ratio de 35.85% est inférieur à ce seuil de 50%, ce qui indique que la taille des tests est plus importante par rapport à la taille du code, reflétant ainsi une bonne quantité de tests. Cependant, il est important de noter que ce ratio ne nous donne pas directement le nombre de tests par classe (TPC). Pour obtenir le TPC, il faudrait diviser le nombre total de tests par le nombre total de classes dans le code.



```
PROBLEMS 1 OUTPUT TERMINAL PORTS DEBUG CONSOLE

Code size to test size ratio for XYSeriesCollection: 58.76%
Code size to test size ratio for LongNeedle: 46.58%
Code size to test size ratio for DialCap: 47.21%
Average code size to test size ratio (M1): 35.85%
PS C:\Users\huang\OneDrive\Documents\GitHub\IFT3913\tp2>
```

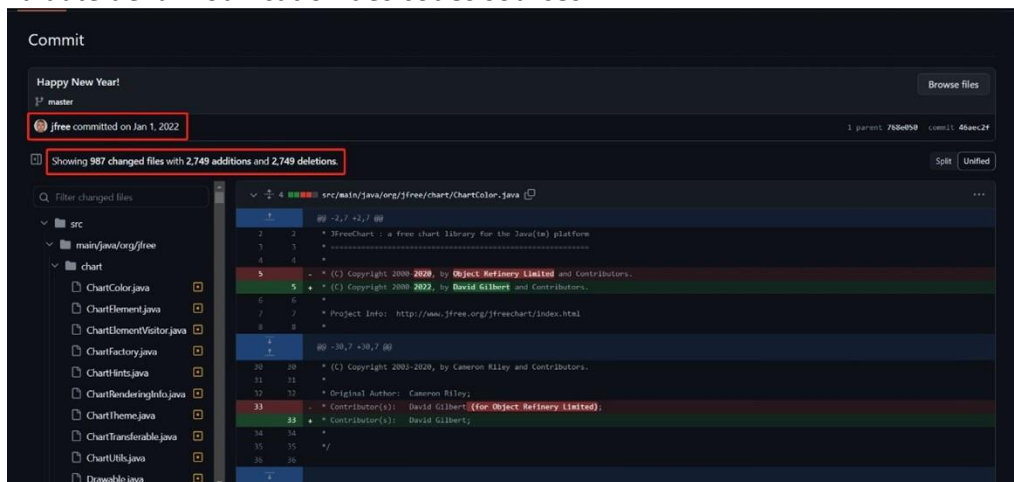
Q2 : Est-ce qu'il y a assez de tests à jour avec le reste du code?

- Métrique M3 : AGE (l'âge d'un fichier) pour les fichiers de test et de code
- Métrique M4 : NCH (nombre de commits dans l'historique d'une classe)

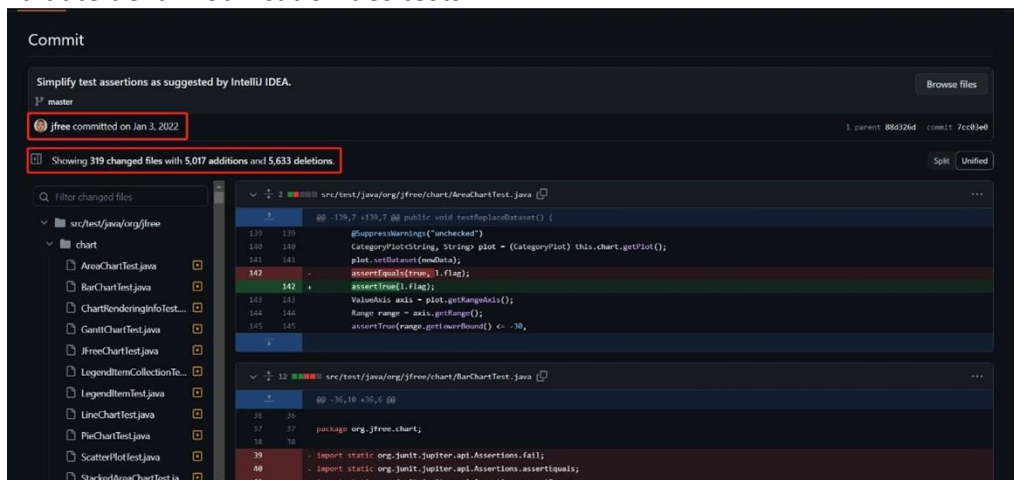
Explication : La métrique M3 compare les dates de dernière modification des fichiers de test et de code - s'il y a une grande différence, cela peut indiquer que les tests ne sont pas à jour. M4 peut indiquer combien de fois une classe a été modifiée, donc si une classe a été fréquemment modifiée mais ses tests pas autant, cela pourrait indiquer que les tests ne sont pas à jour.

Résultats : Pour mesurer le métrique M3 (l'âge des fichiers), on a consulté sur Github les dates de dernière modification des de test et de code de sous fichier 'jfreechart/src/test/java/org/jfree/chart/title'. On remarque la date de la dernière modification est le 1 Janvier 2022 pour le code et 3 Janvier 2022 pour les tests. Il y a une différence de 2 jours entre ces modifications et on considère qu'il y a assez de tests à jour avec le reste du code si l'écart est moins d'une semaine. D'autre part, on remarque qu'il y a 26 commits pour les codes 17 commits pour les tests dans la classe 'title'. Il s'agit d'un écart de 35 %, ce qui est considéré assez de commits dans l'historique, car les codes et les tests ont été fréquemment modifiés.

La date de la modification des codes sources :



La date de la modification des tests:

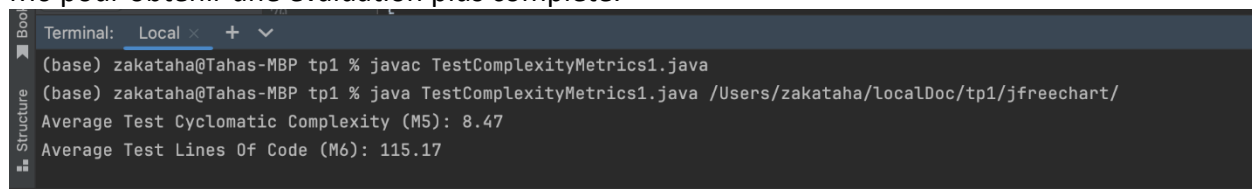


Q3 : Est-ce que les tests sont trop complexes?

- Métrique M5 : CC (Complexité cyclomatique d'une méthode) pour les méthodes de test
- Métrique M6 : LOC (Lines Of Code) pour les fichiers de test

Explication: M5 donne une idée de la complexité des méthodes de test. Un CC élevé pour des tests pourrait indiquer une complexité excessive. M6 donne une idée de la longueur et de la complexité potentielle des fichiers de test.

Résultats : D'après les données recueillies, la complexité cyclomatique moyenne des tests (M5) est de 8.47 et le nombre moyen de lignes de code pour les fichiers de test (M6) est de 115.17. Selon les valeurs de référence pour la complexité cyclomatique définies par Carnegie Mellon¹, une complexité cyclomatique de 1 à 10 indique que les méthodes sont généralement simples et faciles à comprendre. Dans le cas présent, une complexité cyclomatique moyenne de 8,47 se trouve dans cette plage Inférieure. On pourrait donc dire que, en se basant uniquement sur le critère de la complexité cyclomatique (M5), les tests ne sont pas trop complexes. Concernant le nombre moyen de lignes de code (M6) pour les fichiers de test, il n'existe pas de plage standard ou accepté pour déterminer si cette valeur est excessive. Cependant, un fichier de test de plus de 100 lignes pourrait être considéré comme assez long et potentiellement complexe. Il est important de rappeler que la complexité dépend également d'autres facteurs, tels que la clarté du code et des commentaires, les pratiques de codage utilisées, etc. En outre, même si en moyenne les tests ne sont pas trop complexes, cela ne signifie pas prioritairement qu'il n'y a pas de tests individuels qui sont excessivement complexes. En somme, bien que la complexité cyclomatique moyenne suggère que les tests ne sont pas trop complexes, le nombre moyen élevé de lignes de code indique une certaine complexité. Il serait donc utile de compléter cette analyse avec d'autres indicateurs de complexité, ainsi que d'examiner la distribution des valeurs de M5 et M6 pour obtenir une évaluation plus complète.



```
Terminal: Local x + v
(base) zakataha@Tahas-MBP tp1 % javac TestComplexityMetrics1.java
(base) zakataha@Tahas-MBP tp1 % java TestComplexityMetrics1.java /Users/zakataha/localDoc/tp1/jfreeschart/
Average Test Cyclomatic Complexity (M5): 8.47
Average Test Lines Of Code (M6): 115.17
```

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

-Métrique M7 : DC (Densité de commentaires) pour les fichiers de test Métrique M8 : CLOC (Lignes de commentaires) pour les fichiers de test

-Métrique M9 : TLOC (Lignes code qui ne sont pas des commentaire) pour les fichiers

Explication: M7, M8 et M9 peuvent montrer si suffisamment de commentaires sont présents dans les fichiers de test, ce qui peut indiquer le niveau de documentation. Un faible DC, CLOC et TLOC pourrait indiquer un manque de documentation.

Résultats : D'après les données recueillies, la densité moyenne de commentaires dans les tests (M7) est de 32,37%. Cette métrique indique la proportion de lignes de commentaire par rapport au nombre total de lignes de code. Une densité de commentaires de 32,37% indique qu'un niveau de lignes dans les fichiers de test sont des commentaires. Il n'y a pas de consensus clair sur ce

¹ : <https://www.codecentric.de/wissens-hub/blog/why-good-metrics-values-do-not-equal-good-quality>

que devrait être la densité de commentaires idéale. Cependant, une densité d'environ 30 % est souvent considérée comme acceptable dans l'industrie du logiciel. C'est parce qu'un code bien écrit devrait en grande partie se documenter lui-même, sans avoir besoin de commentaires excessifs. Sur cette base, on pourrait dire que les tests sont suffisamment documentés. Toutefois, la quantité de commentaires ne donne pas nécessairement une image complète de la qualité de la documentation. Par exemple, les commentaires peuvent être déroutants, trompeurs ou obsolètes. Il est également important de considérer le type de commentaires - par exemple, s'ils identifient correctement l'intention des tests et les conditions d'entrée et de sortie. Pour une évaluation plus précise de la documentation des tests, il serait donc utile de compléter cette analyse avec une évaluation qualitative des commentaires.

```
Terminal: Local x + v
(base) zakataha@Tahas-MBP tp1 % javac TestDocumentationMetrics.java
(base) zakataha@Tahas-MBP tp1 % java TestDocumentationMetrics.java /Users/zakataha/LocalDoc/tp1/jfreechart/
Average Test Comment Density (M7): 32.37%
(base) zakataha@Tahas-MBP tp1 %
```