IFT3913 QUALITÉ DE LOGICIEL ET MÉTRIQUES – AUTOMNE 2019 – TRAVAIL PRATIQUE 1

Dans ce TP vous allez mesurer la sous-documentation du code java. En autres termes, votre programme va trouver les cas où les développeurs ont créé du code compliqué sans un niveau de documentation adéquat.

PART 1 (40%)

Écrivez un programme qui, étant donné le fichier source d'une classe java, calcule les métriques :

- cLOC : nombre de lignes de code d'une classe
- mLOC : nombre de lignes de code d'une méthode
- cCLOC : nombre de lignes de code d'une classe qui contiennent des commentaires
- mCLOC : nombre de lignes de code d'une méthode qui contiennent des commentaires
- cDC : densité de commentaires pour une classe : cDC = cCLOC / cLOC
- mDC : densité de commentaires pour une méthode : mDC = mCLOC / mLOC

Précisions :

1) Vous devez traiter tous les trois types de commentaires de Java (//, /* */, /** */). Vous devez aussi traiter correctement le cas des commentaires imbriqués.
2) Les lignes vides ne doivent pas être prises en compte
3) Une ligne qui contient à la fois du code et de commentaires (voir exemples au-dessous) compte à la fois pour LOC et CLOC. Exemples :
    - `i++; //increment`
    - `for (String element : theArray) /* guaranteed to contain 2 elements */ {`
4) Vous devez considérer les lignes de javadoc d'une classe ou d'une méthode comme faisant partie du compte total. (NB : Le javadoc d'un classe/méthode, précède le code de la classe/méthode.)

PART 2 (20%)

Faites que votre code prenne en entrée le chemin d'accès d'un dossier qui contient du code java et produise deux fichiers au format CSV (« *comma separated values* », valeurs séparées par des virgules).

1) Fichier classes.csv
    - La première ligne doit être : `chemin, class, cLOC, cCLOC, cDC`
    - Les lignes suivantes doivent afficher les données appropriées pour chaque classe.
2) Fichier methodes.csv[1]
    - La première ligne doit être : `chemin, class, methode`[1]`, mLOC, mCLOC, mDC`
    - Les lignes suivantes doivent afficher les données appropriées pour chaque méthode (quelle que soit la visibilité) de chaque classe.

Précisions : Votre code doit être capable de traiter le dossier récursivement. Par exemple, si li dossier contient un projet java, il devrait produire des résultats pour tous les fichiers java indépendamment de l'endroit où ils se trouvent dans la hiérarchie des dossiers du projet. Traitez les interfaces, les énumérations et les classes abstraites comme des classes.

[1] Pour faciliter la correction, merci d'éviter l'utilisation des accents.

PART 3 (30%)

Ajoutez de fonctionnalité afin que votre code calcule les métriques :

- CC : complexité cyclomatique de McCabe pour chaque méthode
- WMC : « *Weighted Methods per Class* », pour chaque classe. C'est la somme pondérée des complexités des méthodes d'une classe. Si toutes les méthodes d'une classe sont de complexité 1, elle est égale au nombre de méthodes.
- cBC : degré selon lequel une classe est bien commentée cBC = cDC / WMC
- mBC : degré selon lequel une méthode est bien commentée mBC = mDC / CC

Les résultats doivent être ajoutés aux fichiers CSV classes.csv et methodes.csv. Aux premières lignes de chaque fichier, utilisez les titres `WMC`, `cBC` et CC, `mBC` respectivement.

PART 5 (10%)

Appliquez votre outil au code du https://github.com/jfree/jfreechart Identifiez les 3 classes et les 3 méthodes les moins bien commentées et proposez des améliorations. Décrivez votre solution dans un fichier de texte brut (TXT).

PRÉCISIONS GLOBALES

Travaillez en équipes de 2. Le TP est dû le **vendredi 4 octobre 2019 23h59** via StudiUM. Aucun retard ne sera accepté. Vous pouvez utiliser n'importe quel langage du JVM (ex. Java, Scala, Jython, Kotlin, Groovy, …). Un membre de l'équipe doit soumettre un fichier ZIP contenant (a) votre **code**, (b) **documentation** (c) un **fichier JAR exécutable** de façon **autonome** (c.-à-d., incluant toutes les librairies que vous pourriez utiliser), et (d) tout autre information pertinente, incluant les noms des 2 membres de l'équipe, en format **TXT**. L'autre membre doit soumettre juste un fichier contentant les noms des 2 coéquipiers. Votre code doit être compilable et exécutable (même s'il peut être manque quelques fonctionnalités). Code qui ne compile ou n'exécute pas sera accordé un 0, donc assurez-vous d'empaqueter toutes les librairies nécessaires. Manque de documentation en ce qui concerne la façon de compiler, d'exécuter **et d'utiliser** votre code, pourrait entraîner une déduction considérable.

BONUS 10%

Utilisez votre outil pour essayer d'améliorer le niveau de documentation dans un projet java open source sur Github (par exemple : https://github.com/topics/java?o=desc&s=forks). Pour qu'un bonus de 10% soit accordé, vous devez me montrer les résultats de l'application de votre outil au projet, le(s) *pull request(s)* soumis (que vous devez avoir soumis avant l'échéance du TP), et je dois être convaincu que cela a été fait de bonne foi et sérieusement.

Un bonus de chocolat vous sera accordé si un de vos *pull request* est accepté par les développeurs du projet. Pour que le bonus soit accordé, vous devez me convaincre qu'un de vos *pull request* a été réellement accepté par les développeurs du projet. Par exemple, ce n'est pas le cas que vous ayez accepté votre propre *pull request* ou que vous soyez ami avec le responsable du projet et que votre ami ait accepté le *pull request* dans le but du bonus du cours 😎. Je considérerai les *pull requests* qui sont acceptés en tout temps avant le 2019-12-20.