IFT 3913 – Qualité du logiciel et métriques
Université de Montréal/DIRO, Automne 2018

Travail pratique #2

Extraction de métriques

1. Objectif

Extraction de métriques à partir de diagrammes de classes UML (voir syntaxe du TP1).

2. Conditions de réalisation

Mêmes groupes que le TP1. Travail à remettre au plus tard **mercredi, le 7 novembre** à 23h55 pour le code via Studium et lors de la séance de démo du lendemain (jeudi 8) pour les rapports.

3. Travail à réaliser

En utilisant le programme Java qui permet de parser les diagrammes du TP1, ajouter une extension qui permet de calculer les métriques décrites dans la section 5. Le programme doit notamment permettre de

- Visualiser **le résultat du calcul** (dans l'interface usager).
- Accéder, à la demande, à **la définition d'une métrique** choisie par l'utilisateur.
- Produire à la demande **un fichier CSV** (séparation par des virgules) contenant pour un schéma, une matrice ayant pour lignes les classes et pour colonnes les métriques. Chaque cellule donne la valeur d'une métrique pour une classe

Une fois le parser prêt, effectuez une analyse de votre propre code. Pour ce faire, un fichier CSV est attendu contenant l'information suivante (entête CSV) :

Chemin	Nom	Taille (# classes)	Taille (NLOC)	Taille (CLOC)	ANA	NOM	NOA	ITC	…

De ce fichier, extrayez la distribution des mesures : pour chaque colonne, donner ses minimum, maximum, tendance centrale et écart type.

Dans le rapport, un paragraphe de cinq à 10 lignes décrivant ces chiffres est attendu.

4. Barème

1. **Rapport de 3 à 10 pages** incluant la conception du programme (le diagramme de classes détaillé, le manuel utilisateur, et l'analyse) 20% et le manuel d'utilisation 10%. La qualité du français sera prise en compte dans la correction.
2. **Interface usager** 20%
3. **Logique et élégance** (programme) 35%
4. **Analyse du code :** 15%

Exigences spécifiques

1. Concernant le rapport

a) Décrire les cas limites
- Décrire l'arborescence de l'archive soumise (sous dossiers, fichiers spécifiques (readme, classe Main, etc.), extensions, etc.)

b) Inclure une section décrivant les tests effectués (cf. 3 - Autres artefacts)

c) Inclure une section décrivant l'analyse de votre code

d) Précisez les points importants du diagramme UML métier

e) Précisez le temps approximatif passé sur le TP

2. Concernant le code

a) Standardisez le code
- Soyez consis.e ! Prendre garde aux méthodes trop longues, aux indirections trop compliquées
- Encapsulez ! Utiliser des classes usagers dans les JList (ou équivalent) et surcharger les toString()
- User du modificateur "static" avec parcimonie

b) Prendre en charge les problèmes
- Robustesse du code (exceptions)
- Information usager (messages)
- Considérez ceux du TP1 :
  1) fichier vide
  2) présence de classes vides
  3) Espaces blancs non conventionnels (';' sur la même ligne, pas de tabulation, absence d'espaces autour des ':')

3. Concernant l'analyse

a) Fournir un CSV contenant les résultats des mesures appliquées à votre propre code

b) Inclure dans le rapport votre analyse en 5 à 10 lignes

4. Autres artefacts

a) Fournir une batterie de tests
  1) Illustrant les cas limites et leur prise en charge
  2) Démontrant la bonne fonctionnalité de l'application

b) Fournir une Javadoc consistante

5. Métriques

Les métriques à calculer pour une classe sont

1. *ANA(ci)* : Nombre moyen d'arguments des méthodes locales pour la classe ci.
2. *NOM(ci)* : Nombre de méthodes locales/héritées de la classe ci. Dans le cas où une méthode est héritée et redéfinie localement (même nom, même ordre et types des arguments et même type de retour), elle ne compte qu'une fois.
3. *NOA(ci)* : Nombre d'attributs locaux/hérités de la classe ci.
4. *ITC(ci)* : Nombre de fois où d'autres classes du diagramme apparaissent comme types des arguments des méthodes de ci.
5. *ETC(ci)* : Nombre de fois où ci apparaît comme type des arguments dans les méthodes des autres classes du diagramme.
6. *CAC(ci)* : Nombre d'associations (incluant les agrégations) locales/héritées auxquelles participe une classe ci.
7. *DIT(ci)* : Taille du chemin le plus long reliant une classe ci à une classe racine dans le graphe d'héritage.
8. *CLD(ci)* : Taille du chemin le plus long reliant une classe ci à une classe feuille dans le graphe d'héritage.
9. *NOC(ci)* : Nombre de sous-classes directes de ci.
10. *NOD(ci) : Nombre de sous-classes directes et indirectes de ci.*

6. Exemple d'interface usager

La figure suivante donne, à titre d'indication seulement, un exemple d'interface graphique permettant de visualiser les résultats du parsing et du calcul de métriques.