**Documentation**

**Critères de la god Class**

**Métrique de cohésion**

- TCC : Tight Class Cohesion, est le nombre de méthodes publiques directement connectées (qui s’appellent directement) au

sein d’une classe, divisé par le nombre maximum de connexions possibles entre

* LCOM1, LCOM2

Il s’implémente comme si dessous.

Prenez chaque pair de méthode dans une classe. Si elles accèdent au variable membres disjointes, incrémenter P de 1. Si elles partagent au moins une variable, incrémenter Q de 1. Le critère s’exprime comme si dessous.

LCOM1 = P - Q ,si P > Q sinon LCOM1 = 0

LCOM1 = 0 indique une classe cohésive

LCOM1 > indique que la classe n’est pas cohésive puisqu’il ny a pas de méthode utilisant les variables memebres.

*A high LCOM1 value indicates disparateness in the functionality provided by the class. This metric can be used to identify classes that are attempting to achieve many different objectives, and consequently are likely to behave in less predictable ways than classes that have lower LCOM1 values. Such classes could be more error prone and more difficult to test and could possibly be disaggregated into two or more classes that are more well defined in their behavior. The LCOM1 metric can be used by senior designers and project managers as a relatively simple way to track whether the cohesion principle is adhered to in the design of an application and advise changes.*

* LCOM4

Est le critère le plus recommendé. LCOM4 mesure le nombre d’ensemble connectés. Un ensemble connecté est un ensemble contenant des varibles membre et des méthodes qui sont liées. Il devrait y avoir qu’un seul ensemble dans une classe s’il y en a plus la classe manque de cohésion.

Quelles méthodes sont liées ? Soient les méthode a et b, elles sont liées si

1. Elles utilisent la même variable membre
2. A appelle b ou b appelle a

* LCOM4=1 indicates a cohesive class, which is the "good" class.
* LCOM4>=2 indicates a problem. The class should be split into so many smaller classes.
* LCOM4=0 happens when there are no methods in a class. This is also a "bad" class.

**Détail d’implémentations :**

https://github.com/cleuton/jqana/wiki/LCOM4-Calculation

**Métrique de complexité**

WMC : Weighted Method Count, est la mesure de la complexité d’une classe. Cette complexité peut être calculée comme étant :

* le nombre de méthodes dans une classe (WMC) Il faut qu’elle soit minimale.
* Ou la complexité cyclomatique

Les limites : 20 ou 50

Détail d’implémentations :

Il faut analyser chaque fonction, operateur et accesseur. Le constructeur et les event handler compte aussi comme méthode. Les méthodes héritées ne comptent pas. Il suffit alors de sommer le nombre de méthodes ou de pondérer les termes de la somme par la complexité cyclomatique de chaque méthode.

**Métrique pour l’encapsulation**

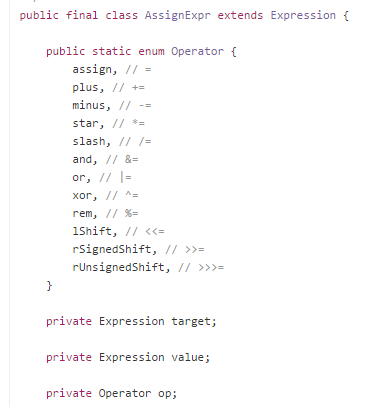
Indicateur du respect de l’encapsulation des autres classes : Access to Foreign Data (ATFD) : représente le nombre de classes extérieures auxquelles la classe accède les attributs (de manière directe ou par les accesseurs). Pas plus que quelques-uns (à définir) comme limite.

**Parser java**

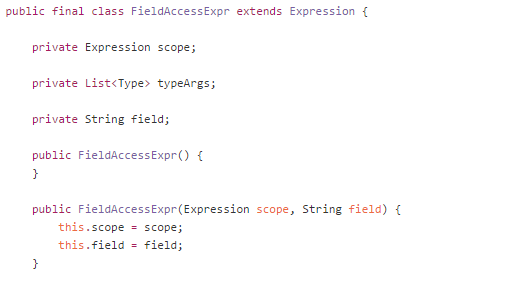
Javadoc : http://www.javadoc.io/doc/com.github.javaparser/javaparser-core/2.4.0

Pour chacune des expressions suivantes il existe une visitor. Pour l’implémenter il faut hériter de la classe VoidVisitorAdapter qui implémente l’interface VoidVistor. Cet interface va contenir la déclaration des visitors de tout les types de nœuds. La liste des nœuds possibles se trouve dans : https://github.com/ftufek/javaparser/tree/master/src/japa/parser/ast/visitor

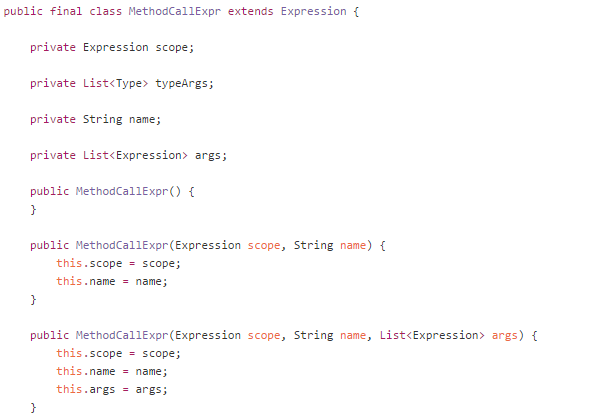
Voici une liste de nœud non exhaustive qui pourrait nous être utiles.



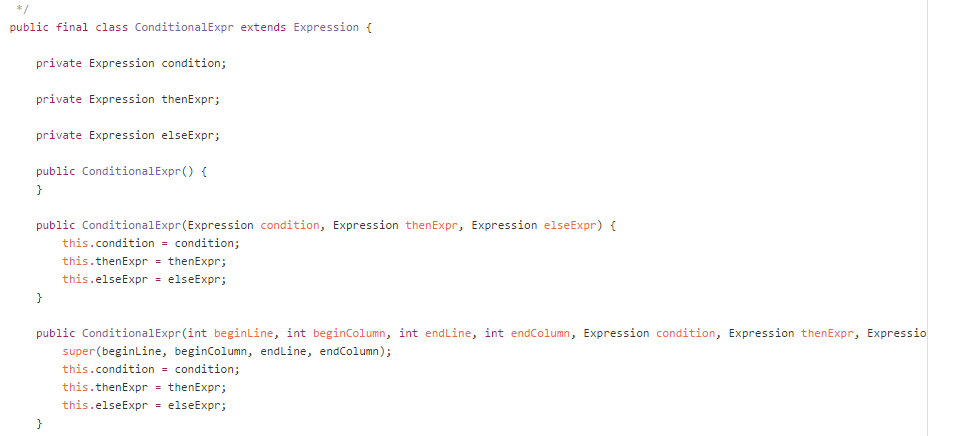
FieldAccessExpr représente les getter et seter d’une class



Les Appels de méthodes



Les conditions



**Junit**

C’est un module intégré dans netbeans. Il sert au TDD (test driven developpement) . Comment ça marche ? Lorsqu’on crée une classe java, on écrit que le squelette de la classe avec ses méthodes. Ensuite on génère, grâce à Junit, une classe test décidé à notre classe. Dans cette classe de test nous retrouvons les méthodes test qui vont instancier notre classe et tester ses méthodes en fonction des paramètres qu’on aura choisis. Ainsi on peut tester les valeurs qui risquent de faire planter notre méthode.

Voici un tutoriel vidéo très bien fait pour JUNIT sous netbeans : <https://www.youtube.com/watch?v=Q0ue-T0Z6Zs>