

UNIVERSITE DE CORSE
2024-2025

Licence ST 3ème année
Option INFORMATIQUE

UE Qualité Logicielle et Tests

CH2 –Métriques logicielles



Evelyne VITTORI
vittori@univ-corse.fr

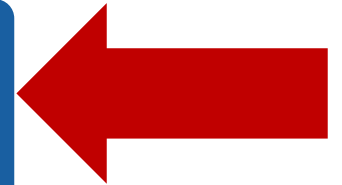
Plan du cours



CH1 – Principes de Qualité



CH2 – Métriques logicielles

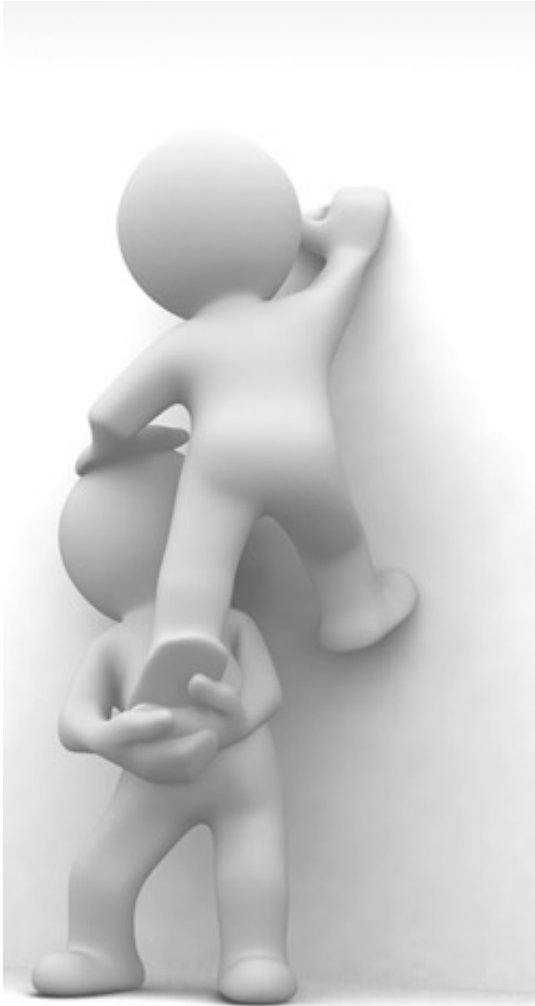


CH3 - Bonnes pratiques OO



CH3 – Tests

Objectifs de ce chapitre



- Comprendre la notion de métrique et son utilité dans le contexte de la qualité.
- Savoir calculer quelques métriques de base pour évaluer la qualité du code.
- Découvrir des outils de calcul automatique des métriques.

CH2- Métriques logicielles

1

Notion de métrique logicielle

2

Métriques de programmation structurée

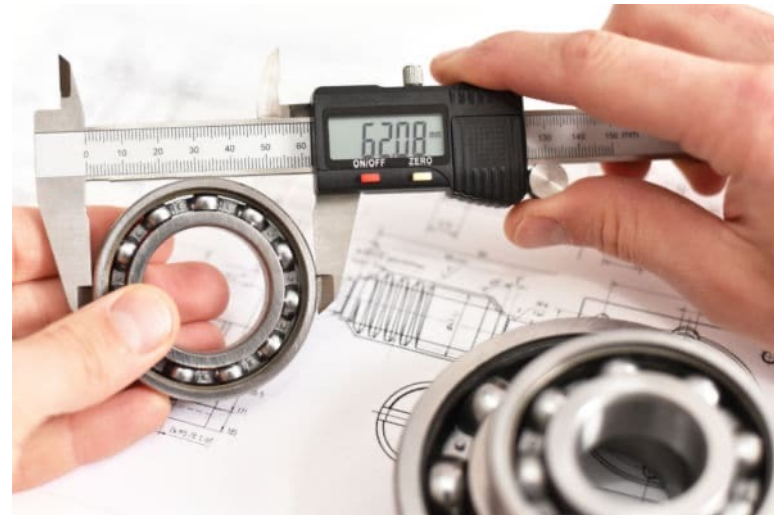
3

Métriques Orientées Objet



*"You cannot control what you cannot
measure"
(DeMarco)*

Notion de métrique logicielle



UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

Qu'est-ce qu'une métrique logicielle (software metric)?

- Un indicateur numérique de **mesure de certaines propriétés** d'un logiciel:
 - Ex: Complexité, couplage, cohésion, ...
- Concept plus ancien que le paradigme OO
 - travaux de recherche dans les années 60
 - 1^{ère} publication 1976: LOC (Lines Of Code), ...
- Différents types de métriques:
 - Métriques logicielles élémentaires
 - Métriques orientées objet

Pourquoi calculer des métriques?

Evaluer la « qualité » d'un logiciel :

- Identifier les points à améliorer
- Limiter les risques d'erreurs
- Faciliter la conception des tests
- Identifier les points de fragilité pour faciliter la maintenance
- Faciliter l'évolutivité du code

Guider l'application des « Best Practices » : acquérir de bons réflexes de conception et structuration du code

Quand calculer des métriques?

- Pendant la phase de développement
 - Amélioration continue
- Avant la phase de tests
- Lors de la maintenance
- Avant d'ajouter de nouvelles fonctionnalités
 - Evaluer l'effort de modification du code en fonction de sa qualité
 - Choix selon le résultat de « l'Audit » du code:
 - Modifications directes
 - Remaniement préalable du code (**refactoring**)
 - Reprogrammation de l'ensemble du code

Catégories de métriques logicielles



3 catégories selon Daskalantonakis*

- Métriques du processus logiciel

- Ex: Détection des défauts dans le développement, coût .

- Métriques du projet

- Ex : nombre de développeurs, efforts effectués dans chaque phase de développement , ratio de réutilisation en conception du logiciel

- Métriques du produit logiciel

- Métriques liées à l'analyse du code
- Ex: taille, compréhensibilité, complexité, structuration, documentation.

Objectif du chapitre

*Michael K. Daskalantonakis. 1992. A practical view of software measurement and implementation experiences within Motorola. IEEE Transactions on Software Engineering, Vol. 18, No. 11.

Les limites des métriques

- Les métriques ne garantissent pas la production d'un « bon » code, elles ne sont qu'une aide.
- Un indicateur pas un **dictateur**!

« Lorsqu'une mesure devient un objectif, elle cesse d'être une bonne mesure »

Loi de GoodHart





Outils de calcul automatiques de métriques



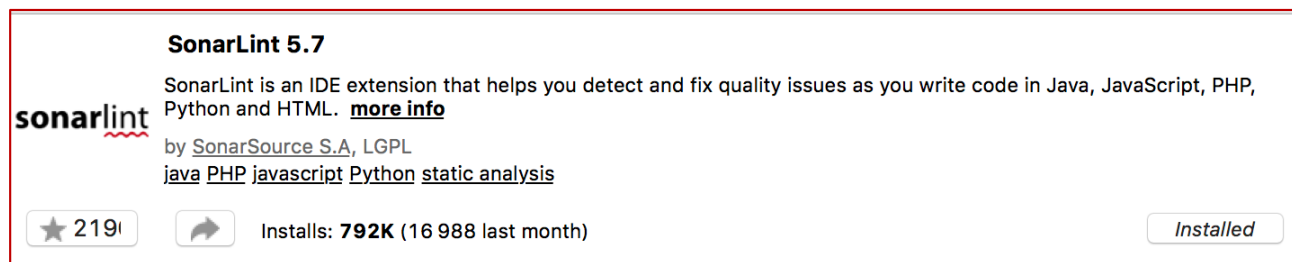
UNIVERSITÀ DI CORSICA
PASQUALE PAOLI

Outils de d'analyse statique

- Deux niveaux d'analyse:
 - Détection des mauvaises pratiques:
 - Non respect des normes et conventions de codage
 - Identification de code dupliqué, code mort....
 - Calcul de métriques
- Outils généralement ajoutés à un IDE sous la forme de plug-in.

SonarLint

- Plugin disponible pour Eclipse, IntelliJ, VSCode
- Analyse du code Java: recherche d'erreurs, respect des conventions
- Installation via Eclipse Marketplace



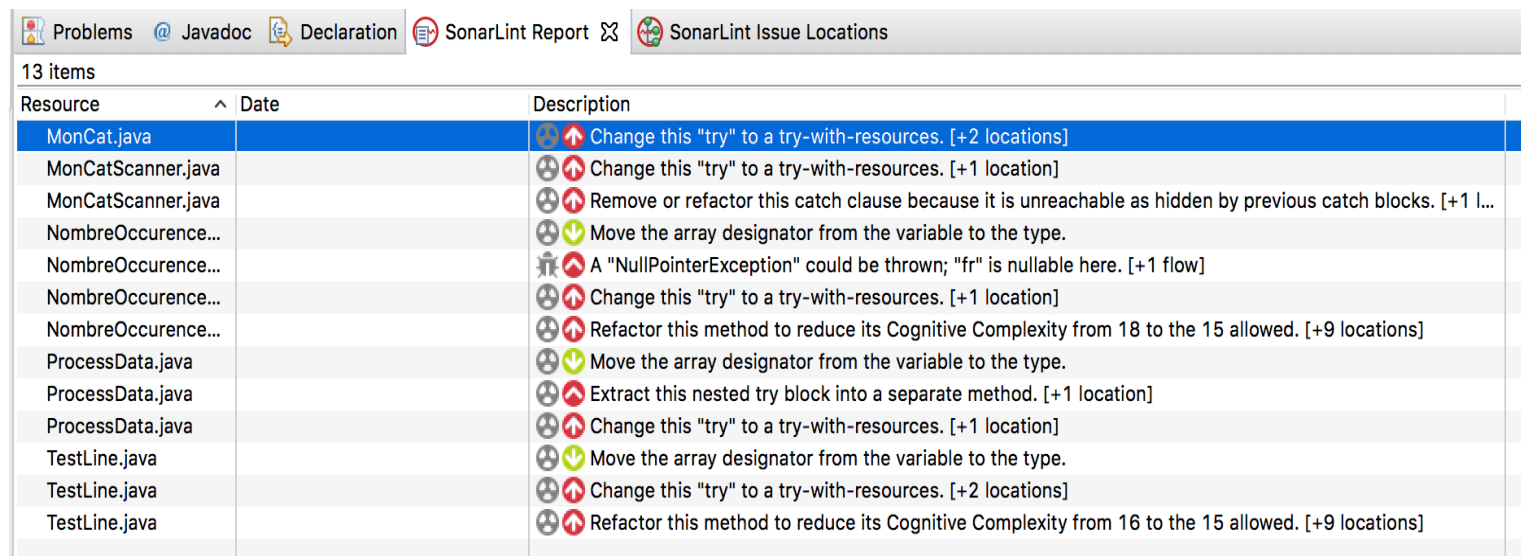
- Plus d'infos sur <https://www.sonarlint.org/>.



SonarLint

Utilisation

- Sélectionnez un projet ou un fichier puis avec le bouton droit choisissez *SonarLint* puis *Analyze*
- Vous devriez voir apparaître la fenêtre de diagnostic de SonarLint.



Problems Javadoc Declaration SonarLint Report SonarLint Issue Locations		
13 items		
Resource	Date	Description
MonCat.java		Change this "try" to a try-with-resources. [+2 locations]
MonCatScanner.java		Change this "try" to a try-with-resources. [+1 location]
MonCatScanner.java		Remove or refactor this catch clause because it is unreachable as hidden by previous catch blocks. [+1 l...
NombreOccurence...		Move the array designator from the variable to the type.
NombreOccurence...		A "NullPointerException" could be thrown; "fr" is nullable here. [+1 flow]
NombreOccurence...		Change this "try" to a try-with-resources. [+1 location]
NombreOccurence...		Refactor this method to reduce its Cognitive Complexity from 18 to the 15 allowed. [+9 locations]
ProcessData.java		Move the array designator from the variable to the type.
ProcessData.java		Extract this nested try block into a separate method. [+1 location]
ProcessData.java		Change this "try" to a try-with-resources. [+1 location]
TestLine.java		Move the array designator from the variable to the type.
TestLine.java		Change this "try" to a try-with-resources. [+2 locations]
TestLine.java		Refactor this method to reduce its Cognitive Complexity from 16 to the 15 allowed. [+9 locations]

Possibilité de paramétrer les règles

Metrics (Plugin Eclipse)

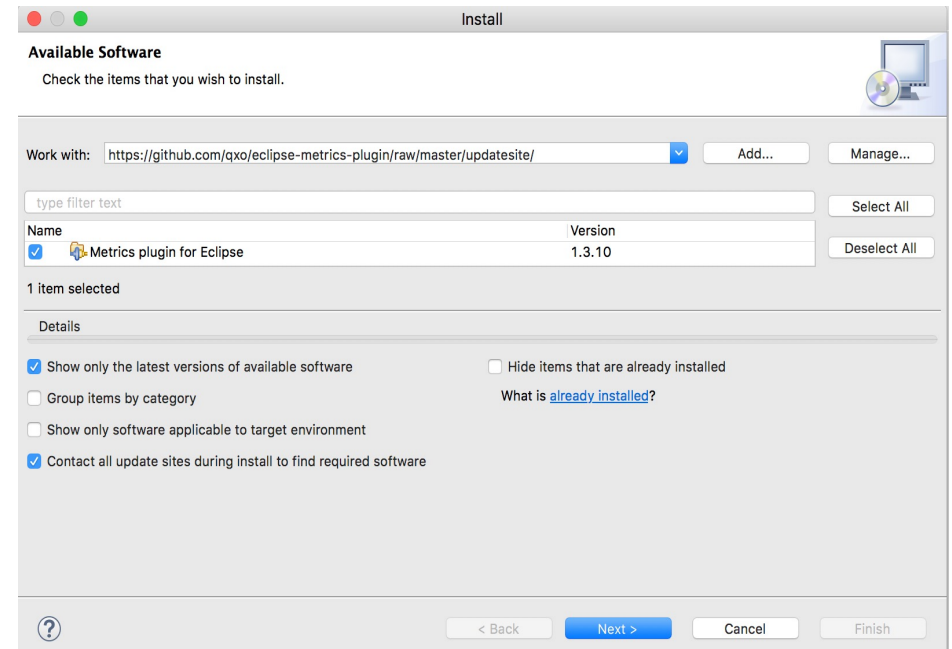
- Le plugin le plus ancien
- Spécialisé dans le calcul de métriques pour du code Java
- 23 métriques mesurées au niveau d'un projet ou d'une classe

Metrics - L3TP5						
Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
▼ McCabe Cyclomatic Complexity (avg/max per...		7,5	3,674	11	/L3TP5/src/util/TestLine.java	main
▼ src		7,5	3,674	11	/L3TP5/src/util/TestLine.java	main
▼ util		7,5	3,674	11	/L3TP5/src/util/TestLine.java	main
▶ TestLine.java		11	0	11	/L3TP5/src/util/TestLine.java	main
▶ NombreOccurenceMot.java		11	0	11	/L3TP5/src/util/NombreOccurenceMot.java	main
▶ MonCat.java		10	0	10	/L3TP5/src/util/MonCat.java	main
▶ ProcessData.java		9	0	9	/L3TP5/src/util/ProcessData.java	main
▶ MonCatScanner.java		9	0	9	/L3TP5/src/util/MonCatScanner.java	main
▶ Translate.java		4	3	7	/L3TP5/src/util/Translate.java	main
▶ LectClavier.java		2	0	2	/L3TP5/src/util/LectClavier.java	main
▶ Number of Parameters (avg/max per method)		1	0	1	/L3TP5/src/util/TestLine.java	main
▶ Nested Block Depth (avg/max per method)		3,375	1,218	5	/L3TP5/src/util/NombreOccurenceMot.java	main
▶ Afferent Coupling (avg/max per packageFrag...		0	0	0	/L3TP5/src/util	
▶ Efferent Coupling (avg/max per packageFrag...		0	0	0	/L3TP5/src/util	
▶ Instability (avg/max per packageFragment)		1	0	1	/L3TP5/src/util	
▶ Abstractness (avg/max per packageFragment)		0	0	0	/L3TP5/src/util	
▶ Normalized Distance (avg/max per packageFr...		0	0	0	/L3TP5/src/util	
▶ Depth of Inheritance Tree (avg/max per type)		1	0	1	/L3TP5/src/util/TestLine.java	
▶ Weighted methods per Class (avg/max per type)	60	8,571	2,871	11	/L3TP5/src/util/TestLine.java	
▶ Number of Children (avg/max per type)	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Overridden Methods (avg/max per...	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Lack of Cohesion of Methods (avg/max per type)		0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Attributes (avg/max per type)	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Static Attributes (avg/max per type)	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Methods (avg/max per type)	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Normal Methods (avg/max per pac...	0	0	0	0	/L3TP5/src/util	
▶ Number of Normal Methods (avg/max per type)	0	0	0	0	/L3TP5/src/util/TestLine.java	
▶ Number of Inherited Methods (avg/max per p...	0	0	0	0	/L3TP5/src/util	

Metrics (Plugin Eclipse)

Installation

- Dans le menu Help, choisissez *Install new software*.
- Dans la zone *work with* saisissez l'adresse <https://github.com/qxo/eclipse-metrics-plugin/raw/master/updatesite/> et validez pour déclencher la recherche
- Décochez la case à cocher "Group Items by categorie" en bas de la fenêtre principale
- Redemarrez Eclipse pour terminer l'installation.



Metrics (Plugin Eclipse)


■ Utilisation du Plug-In

- Sélectionnez un projet et ouvrez la fenêtre de ses propriétés (click droit). Vous devriez voir apparaître Metrics. Sélectionnez cette option et cochez la case "Enable Metrics".
- Dans le menu Window, choisissez *Show view* puis *Other*. Enfin dans le dossier *Metrics*, sélectionnez la vue *Metrics view*.
- Il faut également relancer la construction du projet : bouton droit *Build Project* pour lancer le calcul des métriques.
- Si votre projet est sélectionné, vous devriez voir apparaître la vue de calcul des métriques dans une nouvelle fenetre en partie basse de l'écran.

Vous pouvez personnaliser l'affichage en sélectionnant les métriques qui vous intéressent.

Plugin Eclipse O3SMeasures

- Un petit plugin simple pour le langage Java
- Calcul de 15 métriques basiques
- Installation via Eclipse MarketPlace



o3smeasures v.2.1.0

o3smeasures is a measure-based tool to evaluating software quality using different source-code properties. The tool provides information on software artifacts... [more info](#)

by [Mariana Azevedo](#), GPL

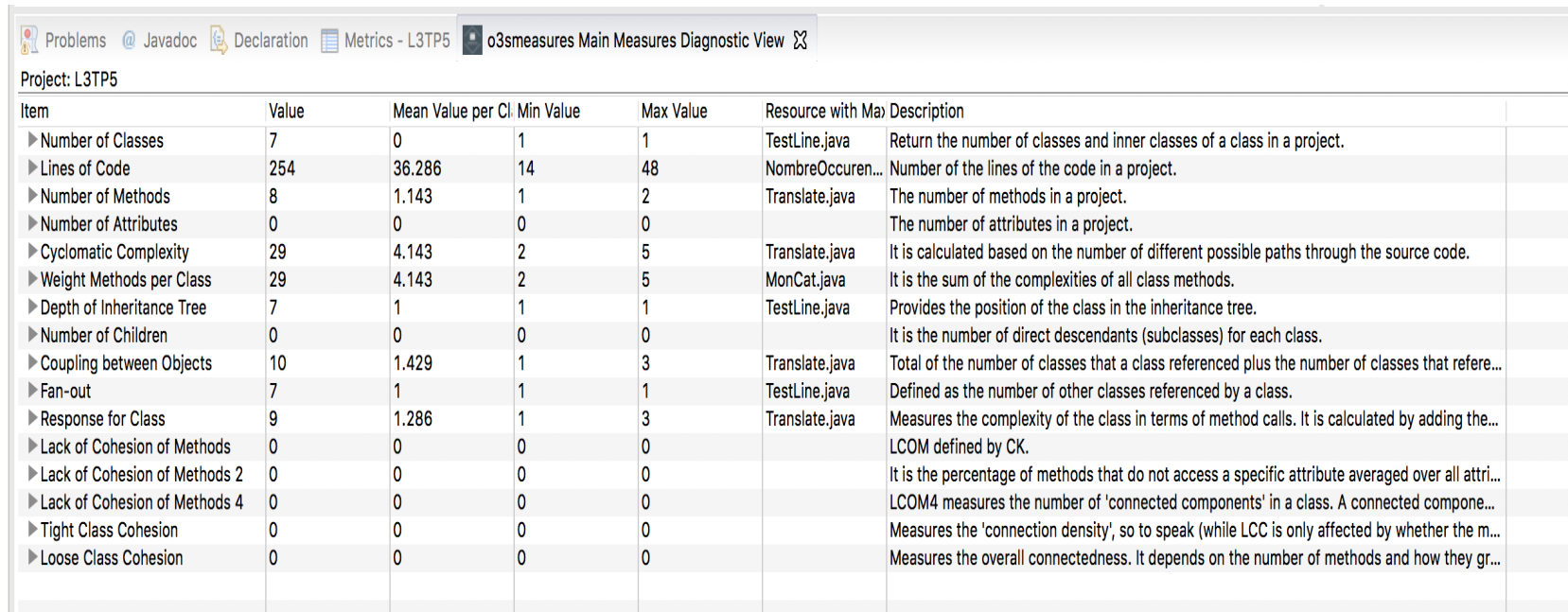
[metrics](#) [software metrics](#) [code quality](#) [quality assessment](#) [Software Quality](#)

- Plus d'informations sur <https://mariazevedo88.github.io/o3smeasures/>.

Plugin Eclipse O3SMeasures

■ Utilisation

- Sélectionnez un projet puis avec le bouton droit choisissez *o3sMeasures* puis *Analyse Java Project*
- Dans le menu Window, choisissez *Show view* puis *Other*. Enfin dans le dossier *o3smeasures*, sélectionnez la vue *o3smeasures main Measure Diagnostic View* pour afficher les valeurs des principales métriques.



Project: L3TP5						
Item	Value	Mean Value per Cl	Min Value	Max Value	Resource with Ma	Description
▶ Number of Classes	7	0	1	1	TestLine.java	Return the number of classes and inner classes of a class in a project.
▶ Lines of Code	254	36.286	14	48	NombreOccuren...	Number of the lines of the code in a project.
▶ Number of Methods	8	1.143	1	2	Translate.java	The number of methods in a project.
▶ Number of Attributes	0	0	0	0		The number of attributes in a project.
▶ Cyclomatic Complexity	29	4.143	2	5	Translate.java	It is calculated based on the number of different possible paths through the source code.
▶ Weight Methods per Class	29	4.143	2	5	MonCat.java	It is the sum of the complexities of all class methods.
▶ Depth of Inheritance Tree	7	1	1	1	TestLine.java	Provides the position of the class in the inheritance tree.
▶ Number of Children	0	0	0	0		It is the number of direct descendants (subclasses) for each class.
▶ Coupling between Objects	10	1.429	1	3	Translate.java	Total of the number of classes that a class referenced plus the number of classes that refere...
▶ Fan-out	7	1	1	1	TestLine.java	Defined as the number of other classes referenced by a class.
▶ Response for Class	9	1.286	1	3	Translate.java	Measures the complexity of the class in terms of method calls. It is calculated by adding the...
▶ Lack of Cohesion of Methods	0	0	0	0		LCOM defined by CK.
▶ Lack of Cohesion of Methods 2	0	0	0	0		It is the percentage of methods that do not access a specific attribute averaged over all attri...
▶ Lack of Cohesion of Methods 4	0	0	0	0		LCOM4 measures the number of 'connected components' in a class. A connected compone...
▶ Tight Class Cohesion	0	0	0	0		Measures the 'connection density', so to speak (while LCC is only affected by whether the m...
▶ Loose Class Cohesion	0	0	0	0		Measures the overall connectedness. It depends on the number of methods and how they gr...

Autres outils d'analyse statique

- CheckStyle

- <https://checkstyle.org/>



- PMD/CPD

- https://pmd.github.io/latest/pmd_userdocs_cpd.html

- FindBugs

- <http://findbugs.sourceforge.net/>



- Wily: métriques pour python

- <https://www.asapdevelopers.com/python-code-complexity/>

- Plus d'outils pour java:

- <https://java-source.net/open-source/code-analyzers>



A faire
tout au
long du
CH2

Exercice : Recherche Outils



- Recherchez des outils de calcul de métriques dans votre IDE préféré (ou ailleurs).
- Proposez des critères de comparaison des outils:
 - Liste des métriques
 - Autres?
- Objectif final:
 - Définition collective d'un tableau comparatif sous Teams