
Analyse de la qualité du code source Java avec JavaNCSS 21.41

Hugo Etiévant

Dernière mise à jour : 20 juin 2004

Qu'est-ce que la qualité ?

La réussite d'un projet de développement nécessite une organisation rigoureuse tant dans la gestion des ressources et des tâches que dans l'organisation technique. Cette dernière implique la définition et le respect de **normes de développement** qui assurent une uniformité du code source en vue d'une maintenance et d'une documentation aisée.

Un code de qualité assure la **pérennité**, la **diffusion** et la **maintenabilité** du projet.

La réussite d'un projet dépend de la qualité du code.

Comment se mesure la qualité ?

Sur quels critères peut-on se baser pour analyser la qualité du code ?

- lisibilité du code
- uniformité des conventions
 - de nommage (variables, méthodes, classes, packages)
 - d'indentation
 - d'organisation (répertoires du projet)
 - de documentation (commentaires, JavaDoc)
- couplage minimum (modules indépendants)
- complexité cyclomatique minimum
- taux de commentaires JavaDoc

Présentation de JavaNCSS

JavaNCSS est un outil développé en Java qui analyse le code source d'un projet Java et calcule des métriques de qualité.

Il est développé par Chr. Clemens Lee

Il est sous licence « GNU General Public License ».

Installation sous Windows

Téléchargement à l'adresse suivante :

<http://www.kclee.com/clemens/java/javancss/>

Décompression dans un répertoire :

[javancss21.41/](#)

Rajout en variable d'environnement de ce répertoire et des archives JAR contenant les classes :

```
set JAVANCSS_HOME=D:\javancss21.41
```

```
set CLASSPATH=%CLASSPATH%;  
%JAVANCSS_HOME%\lib\javancss.jar;  
%JAVANCSS_HOME%\lib\ccl.jar;  
%JAVANCSS_HOME%\lib\jhbasic.jar
```

Mise en œuvre

Une interface graphique permet de visualiser les métriques calculées sur les sources d'un projet.

Syntaxe :

`javancss [-<option>] <sources>`

Le chemin des sources à analyser est défini par `<sources>`. Les options peuvent être les suivantes : `gui` pour lancer l'interface graphique, `xml` pour une sortie XML (texte par défaut), `out <fichier>` spécifie le chemin du fichier de sortie, `recursive` pour un parcours récursif des sous-répertoires, `package` pour une analyse agrégée sur les packages, `object` pour une analyse des classes, `function` pour une analyse des méthodes.

Exemple :

`javancss -gui -recursive D:\projet\src`

Métriques (1)

Métriques sur les packages :

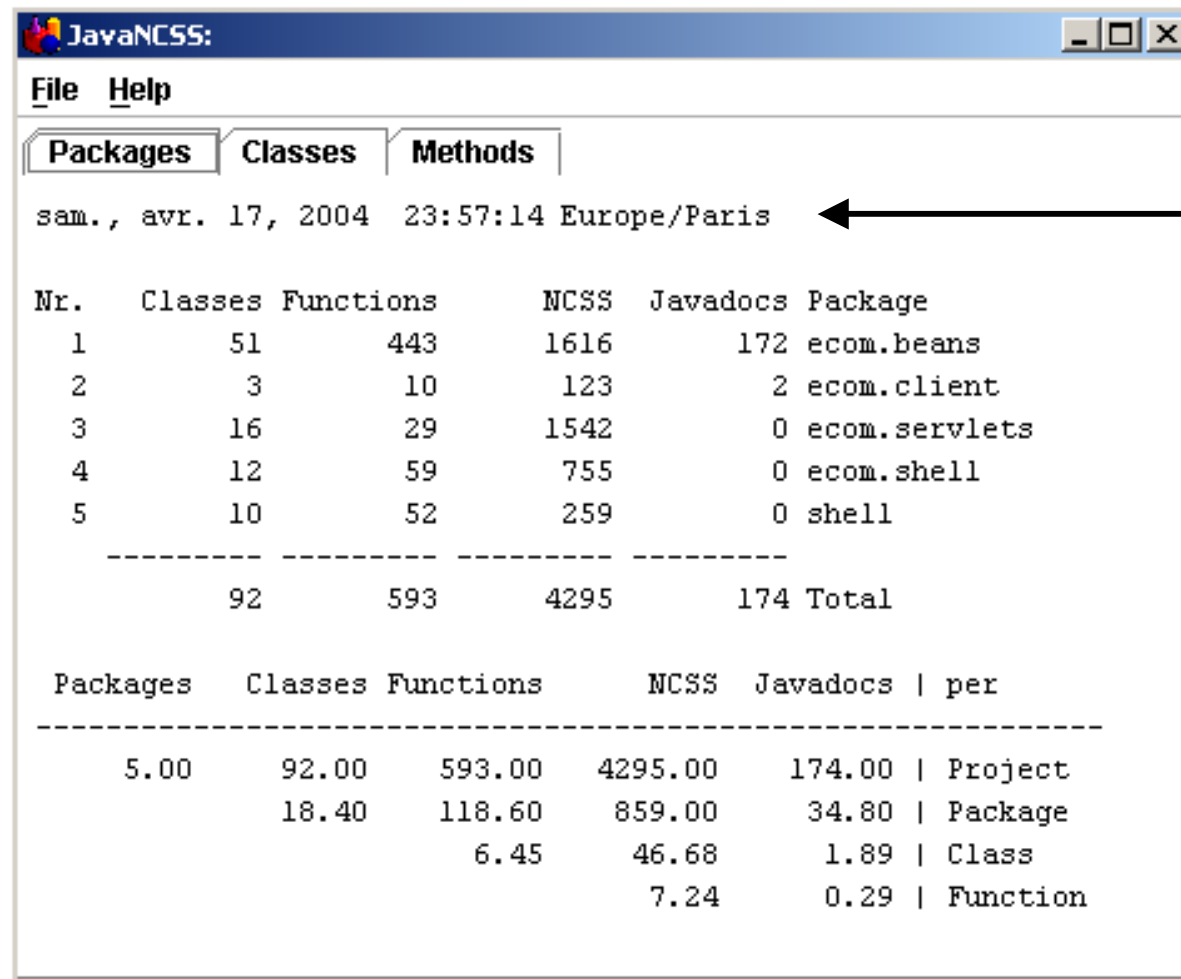
- nombre de classes
- nombre de méthodes
- nombre d'instructions sans commentaire (Non Commenting Source Statements : NCSS)
- nombre de commentaires JavaDoc

Métriques générales :

- nombre de package
- nombre de classes
- nombre de méthodes
- nombre d'instructions sans commentaire
- nombre de commentaires JavaDoc

par projet, package, classe et méthode.

Métriques (2)



The screenshot shows the JavaNCSS application window. The title bar is 'JavaNCSS:'. The menu bar has 'File' and 'Help'. There are three tabs: 'Packages', 'Classes', and 'Methods'. The 'Packages' tab is selected. The main area displays the date and time of analysis: 'sam., avr. 17, 2004 23:57:14 Europe/Paris'. Below this is a table of metrics for five packages. The table has columns: 'Nr.', 'Classes', 'Functions', 'NCSS', 'Javadocs', and 'Package'. The data is as follows:

Nr.	Classes	Functions	NCSS	Javadocs	Package
1	51	443	1616	172	ecom.beans
2	3	10	123	2	ecom.client
3	16	29	1542	0	ecom.servlets
4	12	59	755	0	ecom.shell
5	10	52	259	0	shell

	92	593	4295	174	Total

Below this table is another table showing metrics per package, class, and function. The columns are: 'Packages', 'Classes', 'Functions', 'NCSS', 'Javadocs', and 'per'. The data is as follows:

Packages	Classes	Functions	NCSS	Javadocs	per
5.00	92.00	593.00	4295.00	174.00	Project
	18.40	118.60	859.00	34.80	Package
		6.45	46.68	1.89	Class
			7.24	0.29	Function

Date et heure
d'analyse

Métriques sur
les packages

Métriques
générales

Métriques (3)

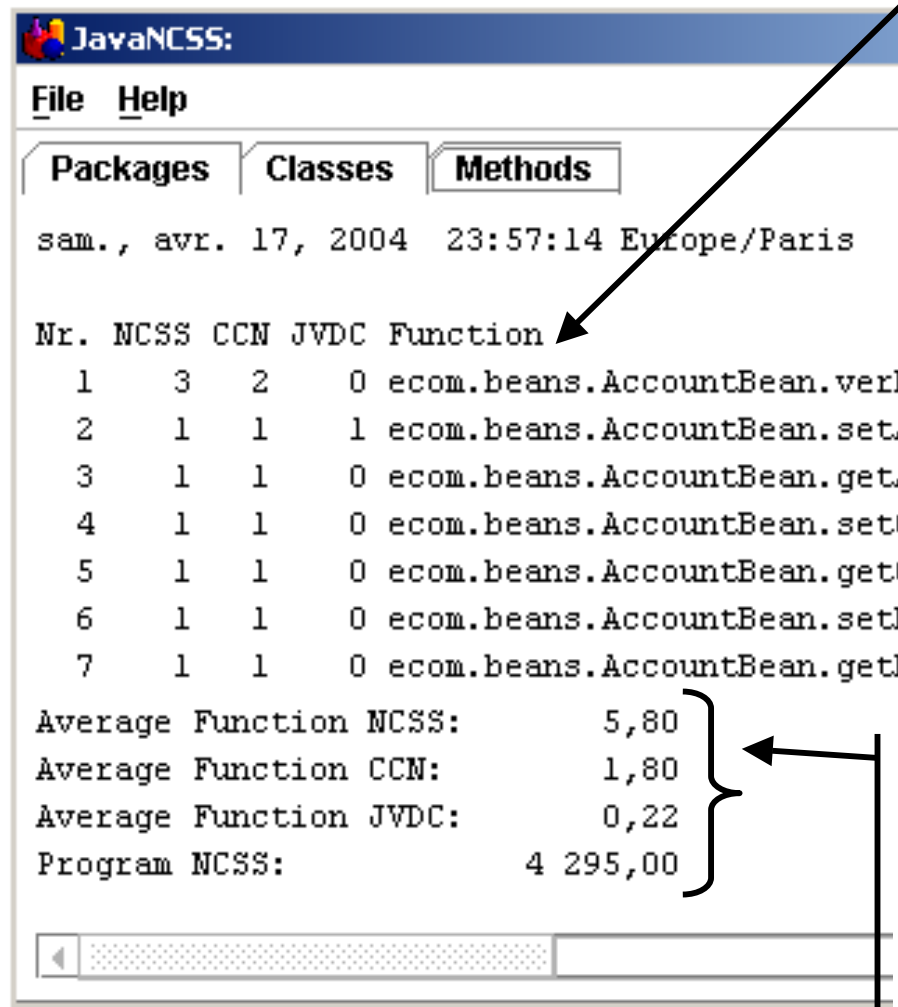
Nr.	NCSS	Functions	Classes	Javadocs	Class
1	50	18	0	7	ecom.beans.AccountBean
2	4	2	0	0	ecom.beans.AccountException
3	4	3	0	4	ecom.beans.AccountLocal
89	6	5	0	0	shell.ShellContext
90	20	6	0	0	shell.ShellContextImpl
91	13	5	0	0	shell.SimpleQuitCommandImpl
92	27	5	0	0	shell.VarCommandImpl

Average Object NCSS: 39,60
Average Object Functions: 6,45
Average Object Inner Classes: 0,00
Average Object Javadoc Comments: 1,89
Program NCSS: 4 295,00

Nombre
d'instructions sans
commentaire, de
méthodes, de sous-
classes, de
commentaires
JavaDocs par classes

Moyennes par classe :
d'instructions sans
commentaire, de
méthodes, de sous-classes,
de commentaires JavaDocs.

Métriques (4)



JavaNCSS:

File Help

Packages Classes Methods

Sam., avr. 17, 2004 23:57:14 Europe/Paris

Nr.	NCSS	CCN	JVDC	Function
1	3	2	0	ecom.beans.AccountBean.verbose(String)
2	1	1	1	ecom.beans.AccountBean.setAccountId(int)
3	1	1	0	ecom.beans.AccountBean.getAccountId()
4	1	1	0	ecom.beans.AccountBean.setOwner(String)
5	1	1	0	ecom.beans.AccountBean.getOwner()
6	1	1	0	ecom.beans.AccountBean.setBalance(double)
7	1	1	0	ecom.beans.AccountBean.getBalance()

Average Function NCSS: 5,80
Average Function CCN: 1,80
Average Function JVDC: 0,22
Program NCSS: 4 295,00

Nombre d'instructions sans commentaire, la complexité cyclomatique (CCN), le nombre de commentaires JavaDocs (JVDC) par méthode.

Moyennes par méthode :
d'instructions sans commentaire,
la complexité cyclomatique (CCN),
le nombre de commentaires
JavaDocs (JVDC).

Exécution avec Ant (1)

Le fichier de déploiement **build.xml** peut être modifié afin de produire automatiquement une analyse de la qualité du code source.

Extraction de la variable d'environnement **JAVANCSS_HOME** :

```
<property name="javancss.home"  
value="${myenv.JAVANCSS_HOME}"/>
```

Les variables suivantes sont nécessaires :

```
<property name="docs.dir" value="docs" />  
<property name="src.dir" value="src" />
```

Exécution avec Ant (2) – format texte

Définition de la tâche :

```
<taskdef name="javancss"  
  classname="javancss.JavancssAntTask"  
  classpath="${javancss.home}"/>
```

Description de la tâche :

```
<target name="javancss" depends="init">  
  <javancss srcdir="${src.dir}"  
    abortOnFail="false" includes="**/*.java" generateReport="true"  
    outputfile="${docs.dir}/javancss_metrics.txt" format="plain"/>  
</target>
```

Cette tâche génère un fichier texte contenant les métriques résultant de l'analyse récursive des fichiers sources « ****/*.java** » contenus dans le répertoire « **srcdir** ». La génération n'est pas stoppée en cas d'erreur (« **abortOnFail** »). Le fichier destination est défini par « **outputfile** » et le format par « **outputfile** ».

Elle dépend de la tâche « **init** » qui crée les répertoires de destination dont celui défini par « **\${docs.dir}** ».

Exécution avec Ant (3) – format XML

Pour générer le rapport au format XML, il suffit de modifier le format de sortie pour « **xml** ».

Description de la tâche :

```
<target name="javancss" depends="init">  
  <javancss  
    srcdir="${src.dir}"  
    abortOnFail="false"  
    includes="**/*.java"  
    generateReport="true"  
    outputfile="${docs.dir}/javancss_metrics.xml"  
    format="xml"/>  
</target>
```

Exécution avec Ant (4) – format HTML

Pour générer le rapport au format HTML, il faut d'abord le générer au format XML puis rajouter une sous-tâche de traduction en HTML via une feuille de style XSLT fournie dans le répertoire **xslt/** de JavaNCSS.

Description de la tâche :

```
<target name="javancss" depends="init">
  <javancss srcdir="${src.dir}" abortOnFail="false"
    includes="**/*.java" generateReport="true"
    outputfile="${docs.dir}/javancss_metrics.xml" format="xml"/>
  <style basedir="${docs.dir}" destdir="${docs.dir}"
    includes="javancss_metrics.xml"
    style="${javancss.home}/xslt/javancss2html.xsl" />
</target>
```

Sortie HTML

Les trois volets de l'interface graphiques sont reproduits en HTML.

La première partie montre les statistiques par packages et celles générales du projet. Les deux autres sont consacrées aux métriques par classes et par méthodes.

Packages

Nr.	Classes	Functions	NCSS	Javadocs	Package
1	51	443	1616	172	ecom.beans
2	3	10	123	2	ecom.client
3	16	29	1542	0	ecom.servlets
4	12	59	755	0	ecom.shell
5	10	52	259	0	shell
	92	593	4295	174	Total

Packages	Classes	Functions	NCSS	Javadocs	per
5.00	92.00	593.00	4,295.00	174.00	Project
	18.40	118.60	859.00	34.80	Package
		6.45	46.68	1.89	Class
			7.24	0.29	Function

Détails sur les métriques

Le nombre d'instructions sans commentaire (Non Commenting Source Statements : **NCSS**) correspond aux commentaires qui auraient du logiquement se trouver dans le code source mais qui font défaut. Il doit être réduit au maximum.

Le nombre de commentaires JavaDoc (**JVDC**) est le nombre de tag JavaDoc trouvés dans le source. Il doit être maximisé.

La complexité cyclomatique (Cyclomatic Complexity Numbers : **CCN**) correspond au degré d'imbrication des structures de contrôle. Il doit être raisonnable (<3) car il n'est jamais bon d'avoir trop de **IF** imbriqués, par exemple (valable aussi pour les **WHILE**, etc...).

Conclusion

Cet outil permet d'obtenir des statistiques détaillées sur le nombre de packages, classes, méthodes par projet.

Il met également l'accent sur le taux de commentaires. Il est un bon indicateur des classes non maintenables : celles possédant les plus grands **NCSS** et **CCN**.

Historique

- ▶ **20 juin 2004** : corrections mineures
- ▶ **18 avril 2004** : création du document (18 diapos)

Agissez sur la qualité de ce document en envoyant vos critiques et suggestions à l'auteur.

Remerciements à Vincent Brabant pour ses remarques.

Pour toute question technique, se reporter au forum Java de Developpez.com

Reproduction autorisée uniquement pour un usage non commercial.



Hugo Etiévant
cyberzoide@yahoo.fr
<http://cyberzoide.developpez.com/>