**Metrics**

Line of code metrics



# Introduction

Is this document I am going to point out some of the metrics of the project and then relate them with code smells.

The metrics were studied in 3 levels:

* Interface
* Class
* Method

# Interface

There are 4 metrics results for each method:

* Comment lines of code (CLOC)
* Javadoc lines of code (JLOC)
* Lines of code (LOC)
* Non-comment lines of code (NCLOC)

We should analyze the CLOC. If we detect an interface with an excessive number of comments in the code, we may have a Comments code smell.

# Class

There are 3 metrics results for each class:

* Comment lines of code (CLOC)
* Javadoc lines of code (JLOC)
* Lines of code (LOC)

The most important ones to analyze and relate to code smells are comment lines of code and lines of code.

A very high percentage of CLOC in total LOC could indicate that there might exist a Comments smell. Jabref has got a class with 824 CLOC and 879 LOC, which mean that approximately 94% of total LOC are CLOC. This does not mean that there is a code smell although it is a good indicator.

A very high number of LOC could also indicate that there might exist a Blob Class. Again, a high number of LOC in a class does not forcibly imply that there is in fact a smell.

# Method:

There are 5 metrics results for each method:

* Comment lines of code (CLOC)
* Javadoc lines of code (JLOC)
* Lines of code (LOC)
* Non-comment lines of code (NCLOC)
* Relative lines of code (RLOC)

The important ones to analyze considering code smells are the CLOC, LOC and RLOC.

If we spot a method with a low CLOC/LOC ratio that might mean that we have a Comments smell. However, this ratio can mean nothing since these comments can be useful.

Methods with a high number of LOC might indicate that we have a long method smell. For example, we have a method with 324 lines which is pretty excessive for a method.