Lines Of Code Metrics

Rodrigo Mesquita 57372

**Metrics:**

* 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)

**Introduction**

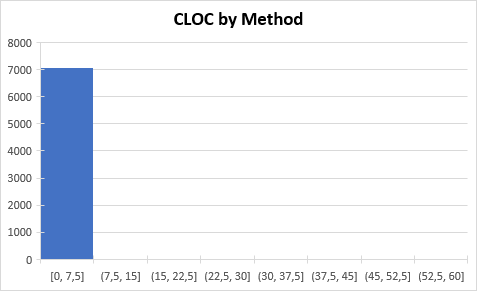
Lines of Code (LOC) metrics’ purpose is to measure the size (number of lines) of files containing a computer programming language. Each metric’s result helps to understand certain aspects from the written code, for example, too large methods and/or classes, classes/methods that are barely commented, etc.

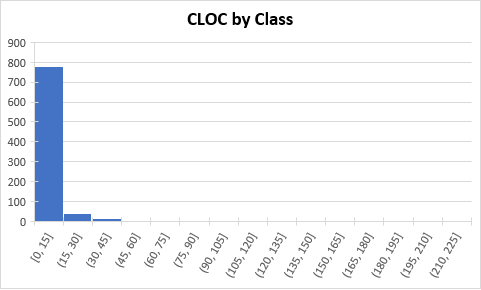
In this document, I will only consider the results obtained through this set of metrics in classes and methods because that is where it would be useful to find one with massive amounts of lines of code or comments.

**Comment Lines of Code** (CLOC)

Calculates the number of lines of code which contain comments. Whitespace lines are not counted.

* It is usually not good to get big numbers here.
* Very useful for classes and methods.





**Thresholds:** By analysing the metric histograms, it’s correct to state that the most common number of comments per method are from 0 to 15. In classes, there are from 0 to 45 comments per class. Thus, our thresholds for methods and classes are 20 and 60, respectively:

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, ecrã

Descrição gerada automaticamenteVery few methods exceed our threshold and should be reviewed (specially the first one) to possibly detect an unnecessarily large comment.

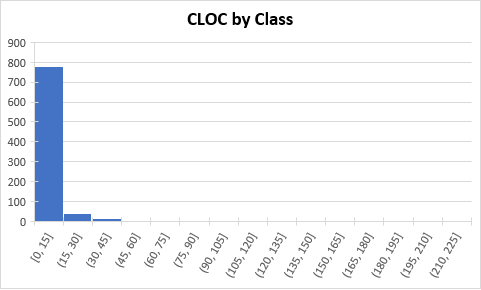
The same goes for some classes that exceeded the threshold.

**Javadoc Lines of Code** (JLOC)

Uma imagem com texto

Descrição gerada automaticamenteCalculates the number of lines of code which contain javadoc comments. Whitespace lines are not counted.

As we can see, the method with the most lines of Javadoc are 20 lines, which in my opinion is not a problem because for some complicated methods it’s for the best interest to have a good and detailed resume about it.



**Threshold:** the most common number of Javadoc lines per class are from 0 to 45, which gives us a threshold of 60.

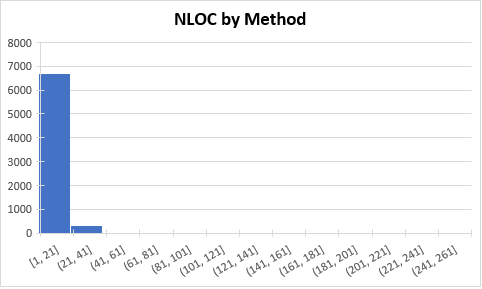
Uma imagem com texto

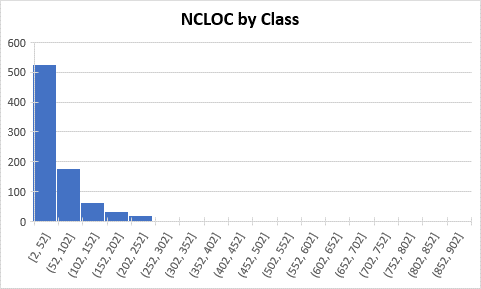
Descrição gerada automaticamente

There are a few excessive cases that can mean the existence of some code smells, for example, large classes. Definitely something that should be checked.

**Lines of Code** (LOC) **and Non-comment Lines of Code** (NCLOC)

The LOC metric seems to be kind of useless in this case since we have already looked into CLOC and JLOC. Thus, I will replace LOC with NCLOC.





**Thresholds:** with the help of the graphs above I will use a threshold of 80 for methods and 400 for classes.

Uma imagem com texto

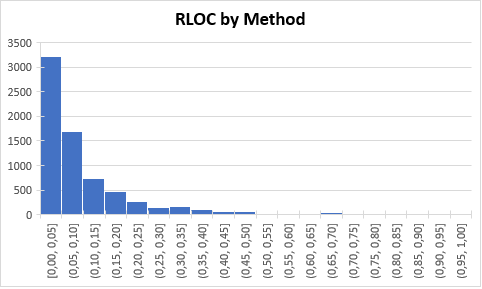
Descrição gerada automaticamenteUma imagem com texto

Descrição gerada automaticamenteIt is undoubtedly necessary to review some methods because they will most likely have a few code smells, especially the long methods one.

My opinion for the classes stands the same as for methods.

**Relative Lines of Code** (RLOC)

Calculates ratio of lines of code for a method to the lines of code for it's containing class, that is, if a method is really close to 100% then the whole class is probably only one method, and such class shouldn’t exist. On the other hand, if a method is 0%, the method is empty and should be removed.



**Threshold:** Due to the histogram results, the threshold for this metric will be 70.

Uma imagem com texto, captura de ecrã, preto, ecrã

Descrição gerada automaticamente

As we can see there are some classes that might only have one method. These classes should be checked and possibly modified.

**Code Smells and Lines of Code metrics**

After using the Lines of Code metrics, I’ve concluded that they are useful to find code smells, for example: long methods, too many comments, no comments, large classes, data classes, speculative generality, etc.

While doing this report I found many classes with no comments, methods with more than 200 lines, classes with more than 1000 lines (with the white spaces lines included), comments that were full methods (probably a reminder or a method to be fixed).

Overall, I’m satisfied with the efficiency of this metrics to find code smells.