**LINES OF CODE**

Purpose of usage:

The chosen metric is a common basis when it comes to estimate either the size or the amount of time/effort that was required to produce a certain software project. This metric can give us an overall idea of the complexity of some of the classes/methods presented in the project. Usually there are big classes/methods which are commonly undesirable as they often times lead to misconceptions and are quite laborious to read. Despite some of the classes/methods being large by default, there are others that can be refactored or even reduced (removing non-functional lines of code). With this metric we can constantly measure the length of classes/methods and carefully plan countermeasures when certain threshold is met.

Looking at the *class metrics table* we can see that the average of the LOC far exceeds the average of the CLOC and JLOC which indicates that there was a moderate amount of documentation produced. After further analyzing the previously mentioned tablewe can state the number of CLOC and JLOC presented in some of the classes are unbalanced, in example, the “*GanttProject”* class is a very dense and complex class that should’ve been more documented. We have to make sure that there is a balance between its documentation and its intricacies.

Notes: LOC -> Lines of Code, JLOC -> Javadoc Lines of Code and CLOC -> Commented Lines of Code.

As stated before, this metric gives us a rough idea of the complexity and code size of the project. Let’s illustrate with some examples/cases:

* If, for instance, we take 2 projects, let’s call them A and B, respectively. The first one being about 10 thousand LOC and the second around 15 thousand LOC, we aren’t able to say much about their relative complexity. On other hand, if we compare project A to a project C that has 300 thousand LOC, in general, the second one is more complex.
* Another thing that we need to consider is the level/experience of the software developer, a more experienced developer tends to write less LOC compared to a less experienced one. This can lead to different values shown in the *class metrics* table.

Taking in consideration the 1st Phase of the Gantt Project we can recall a couple of code smells that are vehemently related to this metric.

Both of the following code smells, “Long Method” and “Reminder” contribute evenly to this metric as they, respectively, increase the number of LOC and CLOC.