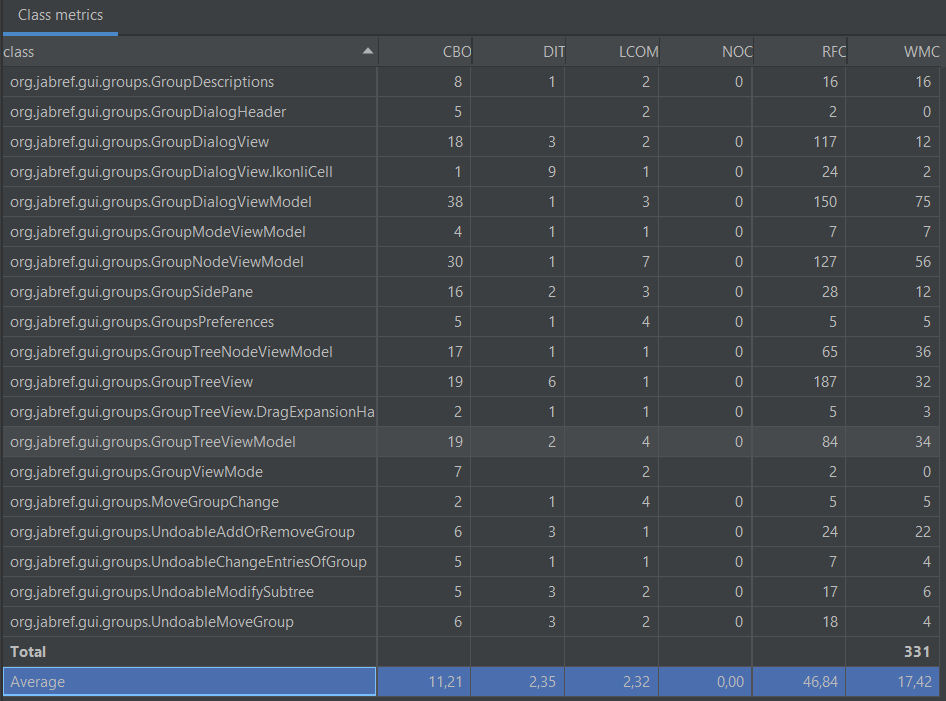
**Chidamber and Kemerer Metrics**

The Chimdamber and Kemerer Metrics consists in the following six metrics calculated for each class:

* Coupling Between Object (CBO) - number of classes to which a class is coupled;
* Depth of inheritance Tree (DIT) - maximum inheritance path from the class to the root class;
* Lack of Cohesion (LCOM) – number of pairs of methods which do not share instance variables, minus the number of pairs of methods that share instance variables of the class;
* Number Of Children (NOC) - number of immediate sub-classes of a class;
* Response For a Class (RFC) – number of methods of a class plus the number of the invoked methods by those class methods;
* Weighted Methods per Class (WMC) – number of methods defined in class.

Extracted values of Chimdamber and Kemerer metrics to the package “groups”.



**CBO** (number of classes to which a class is coupled)

The higher the value of CBO metrics is in a class, the higher is the difficult to reuse it in another app, and harder to test due to the complexity.

The value of CBO in a class should not be higher than 14.

Checking the results of CBO values above, we can see that in the class “GroupsDialogViewModel” the CBO value is 38, picking this as an example, this value is too high. The reuse of this class will be complicated.

**DIT** (maximum inheritance path from the class to the root class)

The deeper is a class in hierarchy, the more methods and variables it will inherit, and due to that, the more complex it becomes.

The value of DIT in a class should not be higher than 5.

If we check the results above, we can find 2 classes that have a higher number than 5 in the DIT value, the class “IkonliCell” and the class “GroupTreeView”, with respectively 9 and 6 DIT values. They become too complex to develop.

**LCOM** (number of pairs of methods which do not share instance variables, minus the number of pairs of methods that share instance variables of the class)

* LCOM = 0

Means that the class is cohesive.

* LCOM > 0

Means that the class needs to be separate into two or more classes.

(Because its variables belong in disjoint sets)

In the results above, all the classes have the LCOM value greater than 0, and the higher this value is, the more likely is this classes to failure.

**NOC** (number of immediate sub-classes of a class)

A high NOC value seems to indicate less problems and indicate a greater reuse of base class, however it will need more testing.

We should be aware that a high value of NOC combined with a high value of WMC can be a sign of poor design due to complexity.

In the results above, all the classes have a NOC value of 0.

**RFC** (number of methods of a class plus the number of the invoked methods by those class methods)

The higher the RFC value is, the more likely to have problems, is very complex and harder to understand.

In the results above we can find some classes with really big numbers of RFC, like “GroupTreeView” or “GroupDialogViewModel” with the values 187 and 150, respectively.

**WMC** (number of methods defined in class)

The higher the WMC value is, the more likely to have problems, the less reusable it gets and the more time it needs to develop and maintain.

A good way to limit the WMC value is that only 10% of the classes can have more than 24 methods.

In the chosen package “groups”, we have 19 classes, 10% of 19 is 1.9, so that means we can only have 1 class with 24 or more methods. As we can see, that doesn’t applies, 5 classes have more than 24 methods.

Checking the class “GroupNodeViewModel”, we can find a ‘Long method’ code smell, assuming that we would solve this problem, we would have to split the method into two or more methods, and that would increase the number of methods defined in class (WMC).