Lcom1

From jpeek JavaDOC

/*LCOM is calculated as the number of pairs of methods operating on disjoint sets of instance variables, reduced by the number of method pairs acting on at least one shared instance variable.

Example:

Say, there are 5 methods in a class.

This means that there are 10 pairs of methods (`5 * 4 / 2`).

Now, we need to see how many of these pairs are using at least one and the same attribute (**Nonempty**) and how many of them are not using any similar attributes (**Empty**).

Then, we just do **`LCOM1 = Empty - Nonempty`**.

The metric can be really big, starting from zero and up to any possible number. The bigger the value the least cohesive is the class.

A perfect design would have `LCOM=0`.

*/

From https://www.aivosto.com/project/help/pm-oo-cohesion.html

LCOM1 Chidamber & Kemerer

LCOM1 was introduced in the Chidamber & Kemerer metrics suite. It’s also called LCOM or LOCOM, and it’s calculated as follows:

Take each pair of methods in the class. If they access disjoint sets of instance variables, increase P by one. If they share at least one variable access, increase Q by one.

LCOM1 = P − Q, if P > Q

LCOM1 = 0 otherwise

LCOM1 = 0 indicates a cohesive class.

LCOM1 > 0 indicates that the class needs or can be split into two or more classes, since its variables belong in disjoint sets.

Classes with a high LCOM1 have been found to be fault-prone.

A high LCOM1 value indicates disparateness in the functionality provided by the class. This metric can be used to identify classes that are attempting to achieve many different objectives, and consequently are likely to behave in less predictable ways than classes that have lower LCOM1 values. Such classes could be more error prone and more difficult to test and could possibly be disaggregated into two or more classes that are more well defined in their behavior. The LCOM1 metric can be used by senior designers and project managers as a relatively simple way to track whether the cohesion principle is adhered to in the design of an application and advise changes.

Lcom3
from javadoc

/*LCOM3 is an attempt to address some shortcomings of the original LCOM1:
LCOM1 gives a value of zero for very different classes -
Its definition is based on method-data interaction, which may not be a correct way to define cohesiveness in the object-oriented world
Very different classes may have an equal value - As LCOM is defined on variable access, it's not well suited for classes that internally access their data via properties
LCOM3 values are in the range [0, 2], where 0 = "high cohesion", 1 = "no cohesion" (class should be split), and values >= 1 suggest serious design flaws in the class, such as unused ("dead") attributes or perhaps the attributes are accessed only from outside the class.
If there are **no more than one method** in a class, LCOM3 is **undefined**.
If there are **no variables** in a class, LCOM3 is **undefined**.
An undefined LCOM3 is displayed as **zero**.
*/
From https://www.aivosto.com/project/help/pm-oo-cohesion.html

Definitions for metric algorithm:

m	number of procedures (methods) in class
a	number of variables (attributes) in class
mA	number of methods that access a variable (attribute)
sum(mA)	sum of mA over attributes of a class

LCOM3 alias LCOM*

$$\mathrm{LCOM3} = (m - \mathrm{sum}(mA)/a) / (m - 1)$$

LCOM3 varies between 0 and 2. Values 1..2 are considered alarming.

In a normal class whose methods access the class's own variables, LCOM3 varies between 0 (high cohesion) and 1 (no cohesion). When LCOM3=0, each method accesses all variables. This indicates the highest possible cohesion. LCOM3=1 indicates extreme lack of cohesion. In this case, the class should be split.

When there are variables that are not accessed by any of the class's methods, 1 < LCOM3 <= 2. This happens if the variables are dead or they are only accessed outside the class. Both cases represent a design flaw. The class is a candidate for rewriting as a module. Alternatively, the class variables should be encapsulated with accessor methods or properties. There may also be some dead variables to remove.

If there are no more than one method in a class, LCOM3 is undefined. If there are no variables in a class, LCOM3 is undefined. An undefined LCOM3 is displayed as zero.