**Lines of Code Metric** João Amorim 57409

**Summary**:

- LOC (Lines of Code) metrics are a quantitative measure used to assess the size and complexity of a software program. They count the number of lines of source code within a program or software project.

- CLOC (Comment Lines of Code): CLOC represents the number of lines in the class that are comments.

- JLOC (Javadoc Lines of Code): JLOC specifically counts lines of code that are part of Javadoc comments in a class.

- LOC (Lines of Code): LOC, as mentioned earlier, represents the total number of lines of code in a class.

**Data Visualization -**

**Top 5 CLOC -**

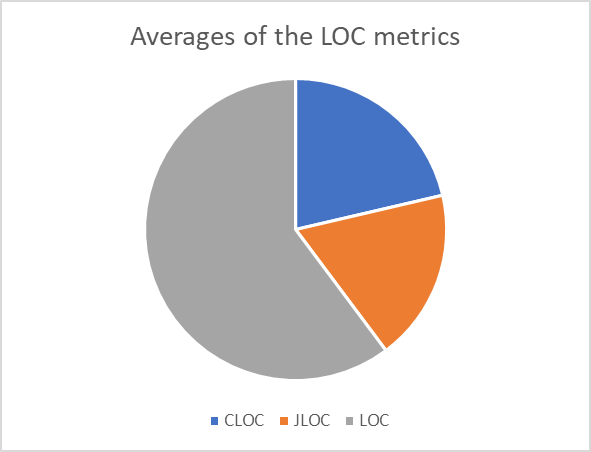
|  |  |  |  |
| --- | --- | --- | --- |
| class | CLOC | JLOC | LOC |
| net.sf.freecol.common.model.Unit | 1962.0 | 1766.0 | 4263.0 |
| net.sf.freecol.common.model.Player | 1921.0 | 1768.0 | 3892.0 |
| net.sf.freecol.client.control.InGameController | 1638.0 | 1297.0 | 4806.0 |
| net.sf.freecol.client.gui.GUI | 1457.0 | 1420.0 | 2220.0 |
| net.sf.freecol.common.util.CollectionUtils | 1445.0 | 1443.0 | 2374.0 |

**Top 5 JLOC -**

|  |  |  |  |
| --- | --- | --- | --- |
| class | CLOC | JLOC | LOC |
| net.sf.freecol.common.model.Player | 1921.0 | 1768.0 | 3892.0 |
| net.sf.freecol.common.model.Unit | 1962.0 | 1766.0 | 4263.0 |
| net.sf.freecol.common.util.CollectionUtils | 1445.0 | 1443.0 | 2374.0 |
| net.sf.freecol.client.gui.GUI | 1457.0 | 1420.0 | 2220.0 |
| net.sf.freecol.client.control.InGameController | 1638.0 | 1297.0 | 4806.0 |

**Top 5 LOC –**

|  |  |  |  |
| --- | --- | --- | --- |
| class | CLOC | JLOC | LOC |
| net.sf.freecol.client.control.InGameController | 1638.0 | 1297.0 | 4806.0 |
| net.sf.freecol.common.model.Unit | 1962.0 | 1766.0 | 4263.0 |
| net.sf.freecol.server.model.ServerPlayer | 1164.0 | 777.0 | 4217.0 |
| net.sf.freecol.common.model.Player | 1921.0 | 1768.0 | 3892.0 |
| net.sf.freecol.server.control.InGameController | 1073.0 | 738.0 | 3451.0 |



**Discussion-**

As it can be seen in the tables our top 5 tables are dominated by the same 5 classes in exception for LOC which means these classes are the most well documented classes, for example we can see this class net.sf.freecol.client.control.InGameController in all of the tables which mean its class of high importance in the code space having the most LOC and being documented with java doc and comments. Now looking at the pie chart which reflects the averages we can that the number of lines of java doc and comments are pretty much the same.

Now these results can be associated with the code smells reported, like the large method and Duplicated code, the class that I mentioned that had the most LOC can be a target of having these code smells because trough out the exploration of the code base I saw big use of java doc in large methods and to explain the steps inside the method was used a lot of comments so that class can be a super class per say, that is doing more than it should be.