Code Metrics

Line of Code (LOC) Metrics

Line of code metrics gives us information about the amount of lines of code that exist in our program. The information is subdivided into sub-metrics that include, packages, methods, classes, interfaces, modules and the lines of codes in the whole project. The lines of codes are divided into different types as well: comments (CLOC), Javadoc (JLOC), non-comment (NCLOC), the percentage of relative code lines (RLOC), and the total lines of code (LOC).

Metrics | Lines of code metrics for Project 'projecto_es' from Tue, ...

Method metrics | Class metrics | Interface metrics | Package metrics | Module metrics | File type metrics | Project metrics

method	CLOC	JLOC	LOC	NCLOC	RLOC
net.sf.freecol.server.model.ServerPlayer.csCombat(FreeColGameObject, Fr	42	9	495	455	11.72%
net.sf.freecol.client.gui.panel.report.ReportCompactColonyPanel.updateCo	68	5	365	298	36.21%
net.sf.freecol.server.ai.ColonyPlan.assignWorkers(List<Unit>, boolean, Lo	93	8	349	256	25.87%
net.sf.freecol.common.model.Map.searchMap(Unit, Tile, GoalDecider, Cost	92	34	329	225	12.97%
net.sf.freecol.server.ai.EuropeanAIPlayer.giveNormalMissions(LogBuilder, L	38	5	281	246	10.62%
net.sf.freecol.common.model.Player.getAllColonyValues(Tile)	56	9	278	226	7.00%
net.sf.freecol.server.ai.REFAIPlayer.giveNormalMissions(LogBuilder, List<A	42	3	276	238	32.55%
net.sf.freecol.server.model.ServerColony.csNewTurn(Random, LogBuilder,	47	9	254	208	31.87%
net.sf.freecol.server.generator.SimpleMapGenerator.makeNativeSettlement	31	8	252	221	25.56%
net.sf.freecol.common.model.Specification.fixDifficultyOptions()	34	9	249	215	8.55%
net.sf.freecol.FreeCol.handleArgs(String[])	19	6	248	236	16.88%
net.sf.freecol.server.model.ServerRegion.requireFixedRegions(Map, LogBui	30	7	243	212	52.37%
net.sf.freecol.common.model.Specification.fixSpec()	48	5	231	183	7.94%
net.sf.freecol.client.gui.menu.DebugMenu.addGameMapOptions(Game, GU	12	0	228	216	66.47%

After some research to find the rule of thumb when it comes to knowing how many lines of code is too much for a certain method or class, I come to the conclusion that over 200 lines for a class is too much and 50-60 lines is the limit for methods. This is to ensure correct usage of certain philosophies that come with programming in OOP.

With this information we can find many cases where these rules are broken and values go way over the supposed limit.

Classes

Method metrics | Class metrics | Interface metrics | Package metrics | Module metrics | File type metrics | Project metrics

class	CLOC	JLOC	LOC
net.sf.freecol.client.control.InGameController	1,638	1,297	4,806
net.sf.freecol.common.model.Unit	1,962	1,766	4,263
net.sf.freecol.server.model.ServerPlayer	1,164	777	4,217
net.sf.freecol.common.model.Player	1,921	1,768	3,892
net.sf.freecol.server.control.InGameController	1,073	738	3,451
net.sf.freecol.common.model.Colony	1,223	1,088	2,818
net.sf.freecol.common.model.Specification	883	676	2,740
net.sf.freecol.server.ai.EuropeanAIPlayer	731	545	2,645
net.sf.freecol.common.model.Tile	1,172	1,025	2,504
net.sf.freecol.common.util.CollectionUtils	1,445	1,443	2,374
net.sf.freecol.client.gui.SwingGUI	742	645	2,321
net.sf.freecol.client.gui.GUI	1,457	1,420	2,220
net.sf.freecol.common.model.Map	899	731	2,099
net.sf.freecol.server.control.InGameControllerTest	125	21	1,833

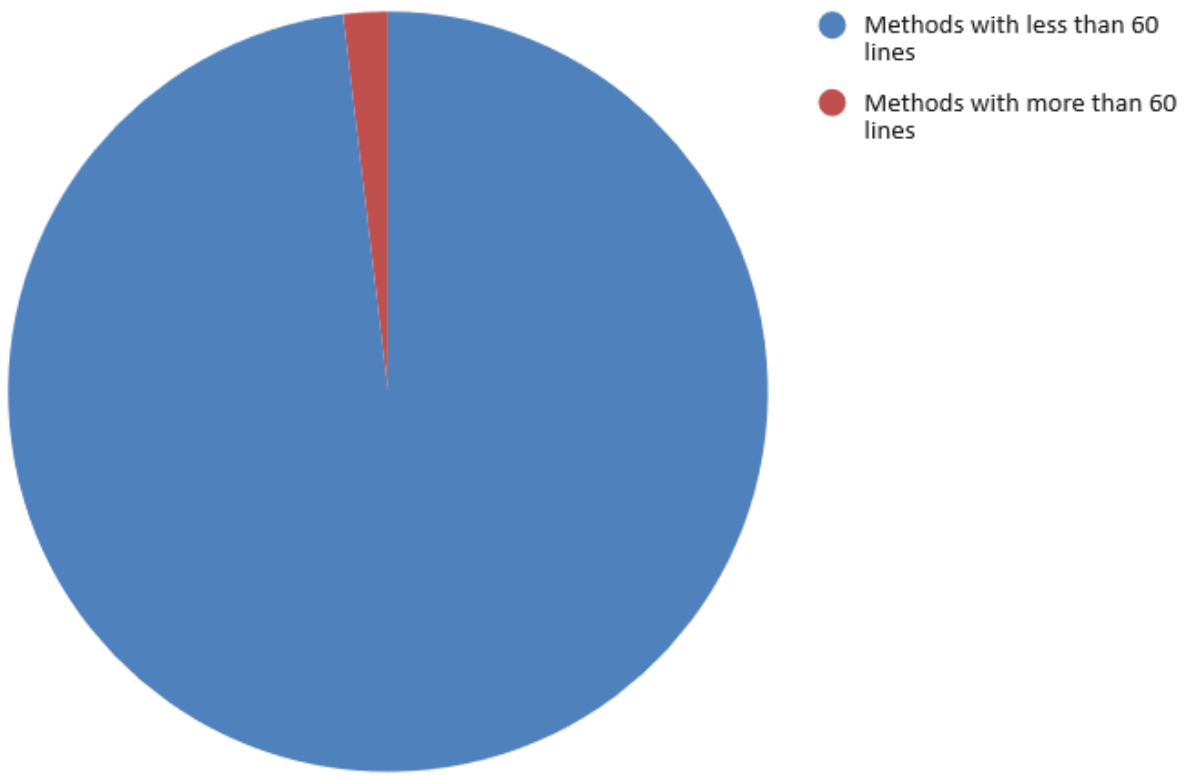
Here we can see that the code has an overwhelming amount of classes with over 2,000 lines of code. This can cause reduced readability and maintainability. It can also be more difficult for debugging and code review.

Methods

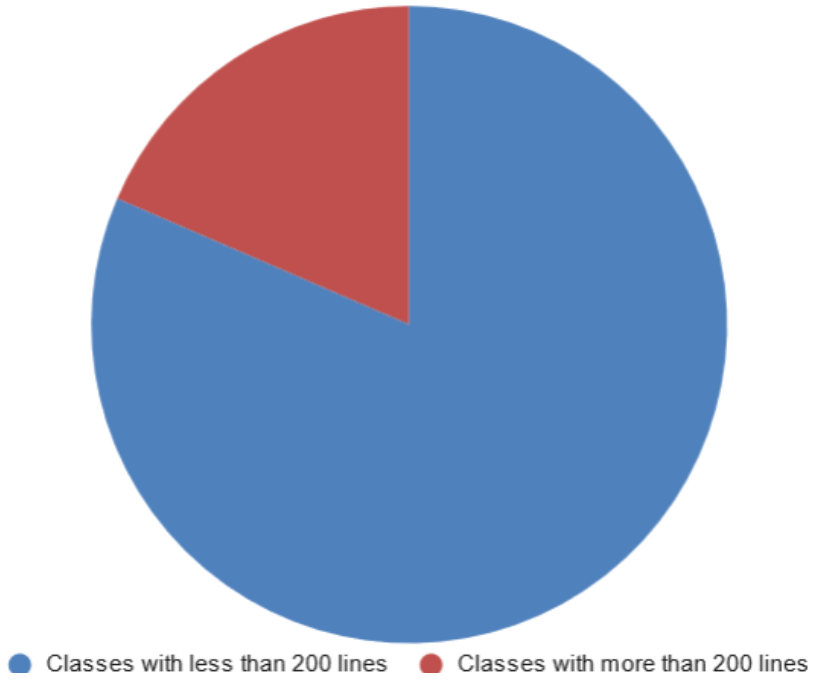
Method metrics | Class metrics | Interface metrics | Package metrics | Module metrics | File type metrics | Project metrics

method	CLOC	JLOC	LOC	NCLOC	RLOC
net.sf.freecol.server.model.ServerPlayer.csCombat(FreeColGameObject, Fr	42	9	495	455	11.72%
net.sf.freecol.client.gui.panel.report.ReportCompactColonyPanel.updateCo	68	5	365	298	36.21%
net.sf.freecol.server.ai.ColonyPlan.assignWorkers(List<Unit>, boolean, Lo	93	8	349	256	25.87%
net.sf.freecol.common.model.Map.searchMap(Unit, Tile, GoalDecider, Cost	92	34	329	225	12.97%
net.sf.freecol.server.ai.EuropeanAIPlayer.giveNormalMissions(LogBuilder, L	38	5	281	246	10.62%
net.sf.freecol.common.model.Player.getAllColonyValues(Tile)	56	9	278	226	7.00%
net.sf.freecol.server.ai.REFAIPlayer.giveNormalMissions(LogBuilder, List<A	42	3	276	238	32.55%
net.sf.freecol.server.model.ServerColony.csNewTurn(Random, LogBuilder,	47	9	254	208	31.87%
net.sf.freecol.server.generator.SimpleMapGenerator.makeNativeSettlement	31	8	252	221	25.56%
net.sf.freecol.common.model.Specification.fixDifficultyOptions()	34	9	249	215	8.55%
net.sf.freecol.FreeCol.handleArgs(String[])	19	6	248	236	16.88%
net.sf.freecol.server.model.ServerRegion.requireFixedRegions(Map, LogBui	30	7	243	212	52.37%
net.sf.freecol.common.model.Specification.fixSpec()	48	5	231	183	7.94%
net.sf.freecol.client.gui.menu.DebugMenu.addGameMapOptions(Game, GU	12	0	228	216	66.47%

In this screenshot, we can see that we have many methods with over 200 lines of code, and most of them have the majority with non-comment lines of code, meaning that most of it is actual code and not simply commenting. Some methods will have many lines of code, such as setting up user interfaces, etc.. But in our example, the method cs.Combat, consisting of 495 total lines of codes, is just to set up combat between an attacker and defender.



Using the metrics to create a pie chart, we can see that there is only a small amount of methods in the full project with more than the ideal 60 lines, but even so, the value is still noticeable.



Using the same metrics to create a pie chart, but this time for the classes. After evaluation we can see that there is a substantial amount of classes with more than the 200 ideal amount, which can cause future problems in debugging and code maintenance.

Relation to Code Smells

The values we find can help us identify code smells that have relation to methods or classes that are too large. This can affect future code refactoring or debugging for future developers. Documenting larger methods or classes can also be more difficult and more susceptible to errors in understanding between documenter and coder.