Group 22B - Assignment 2

Task 1

Class level metrics:

We will be using the following class level metrics to identify which classes should be refactored:

ATFD - Access to Foreign Data:

As per - M. Lanza, R. Marinescu. Object-Oriented Metrics in Practice. Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer-Verlag Berlin Heidelberg, 2006. The desirable value of this metric is less than 6.

CBO - Coupling Between Objects:

According to - R. Shatnawi, W. Li, J. Swain, and T. Newman. Finding software metrics threshold values using ROC curves. J. Soft. Maint. Evol., 22(1):1–16, 2010. The threshold value of this metric is 13.

NPA - Number of Public Attributes:

As per - M. Lanza, R. Marinescu. Object-Oriented Metrics in Practice. Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer-Verlag Berlin Heidelberg, 2006. Desirable value of this metric is less than 4.

RFC - Response for a Class:

According to - R. Shatnawi, W. Li, J. Swain, and T. Newman. Finding software metrics threshold values using ROC curves. J. Soft. Maint. Evol., 22(1):1–16, 2010. The threshold value of this metric is 44.

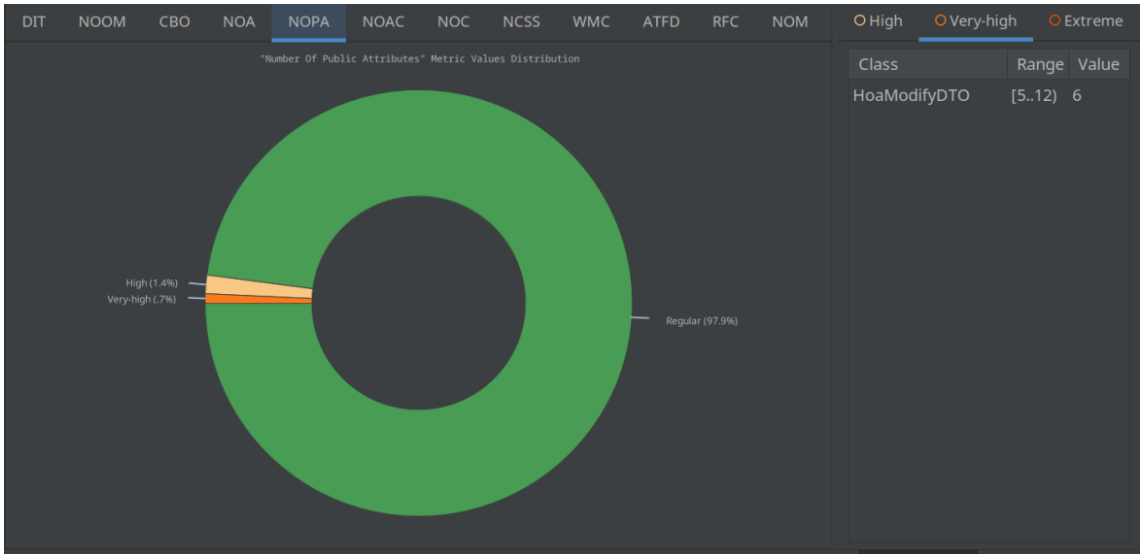
WMC - Weighted Methods per Class:

In line with - Filó, T. G., Bigonha, M., & Ferreira, K. 2015. A catalogue of thresholds for object-oriented software metrics. Proc. of the 1st SOFTENG, 48–55.

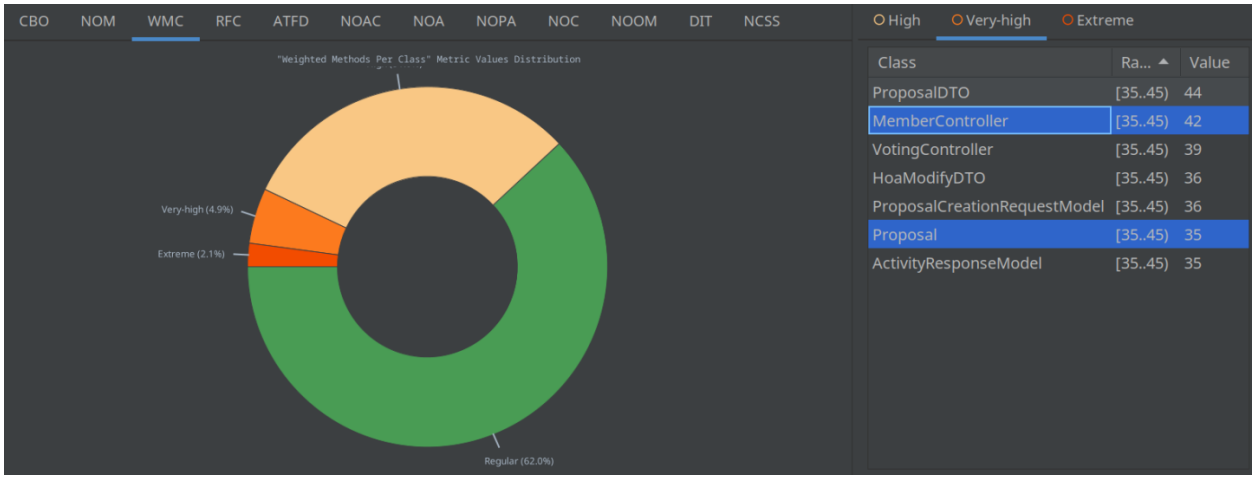
Common value for this metric is less than 12, casual - between 12 and 34, uncommon is greater than 34. Hence we'll be taking the threshold to be 34.

Class	ATFD	CBO	DIT	NCSS	NOAC	NOA	NOC	NOM	NOOM	NOPA	RFC	TCC	WOC	WMC
VotingController	10	30	1	146	0	2	0	9	0	0	83	0.75	1.0	39

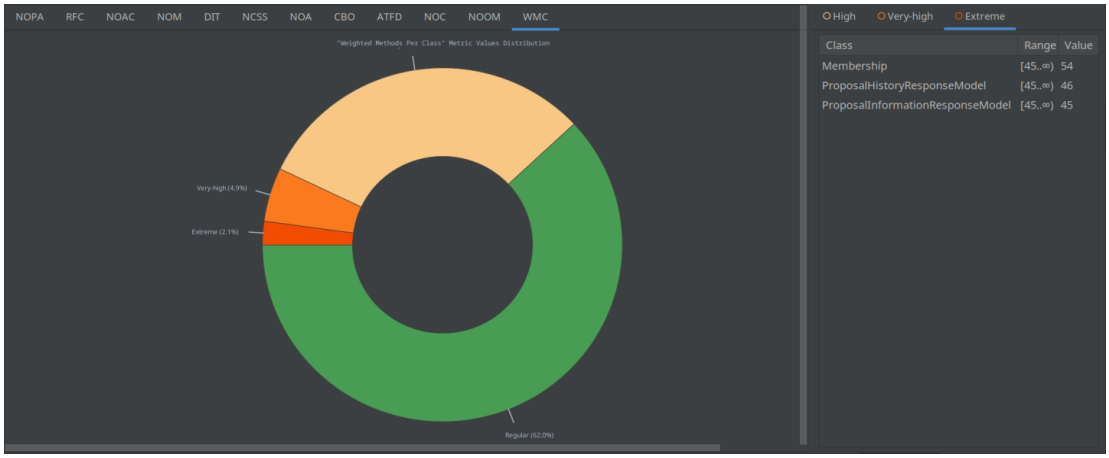
We found one large class (**VotingController**) which fell into the extreme category for multiple metrics, namely: **ATFD**, **CBO** and **RFC**.



Using the **NPA** metric, we have identified a class which goes over the thresholds and hence discovered that **HoaModifyDTO** has getters and setters for every attribute, while also having all its attributes set to public and accessed directly from many methods.



Using the weighted methods per class metric, we have identified two classes which go over the threshold (>34), namely **MemberController**, and **Proposal**.



As we've been recently informed, lombok annotations should not be regarded as code smells, therefore we have replaced the below three methods with other refactoring above. We decided to leave this in the document nonetheless, but we do not count it towards the 5 classes refactored.
Using the weighted methods per class metric, we have identified three classes in the extreme category (>45) that needed refactoring. The **Membership** class and the response models **ProposalHistory** and **ProposalInformation**.

Method level metrics:

We will be using the following class level metrics to identify which methods should be refactored:

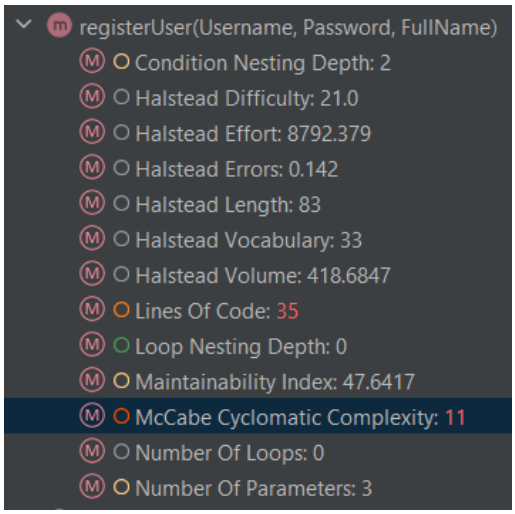
Lines of code (LOC):

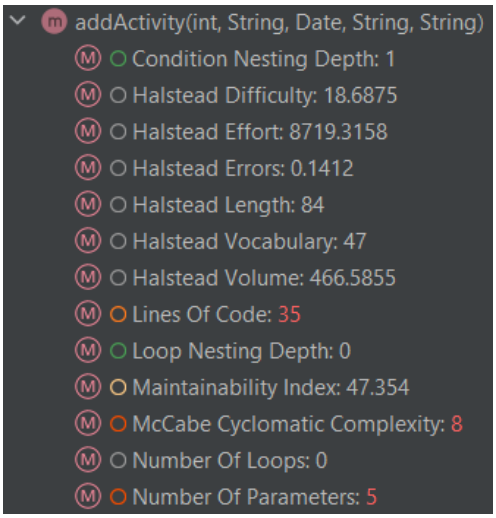
According to Filó, T. G., Bigonha, M., & Ferreira, K. 2015. A catalogue of thresholds for object-oriented software metrics. Proc. of the 1st SOFTENG, 48–55. The desired metric is 30.

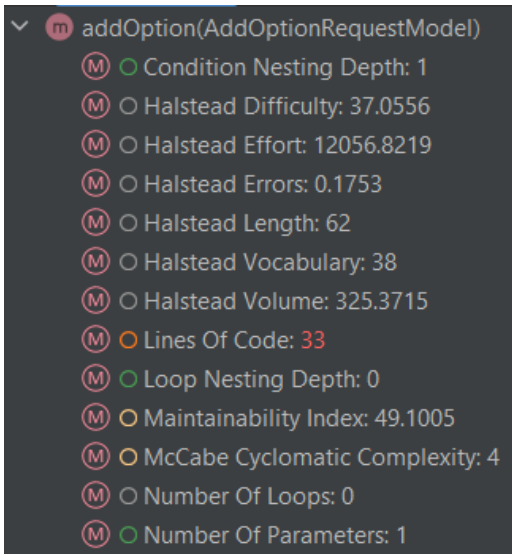
McCabe Cyclomatic Complexity (CC):

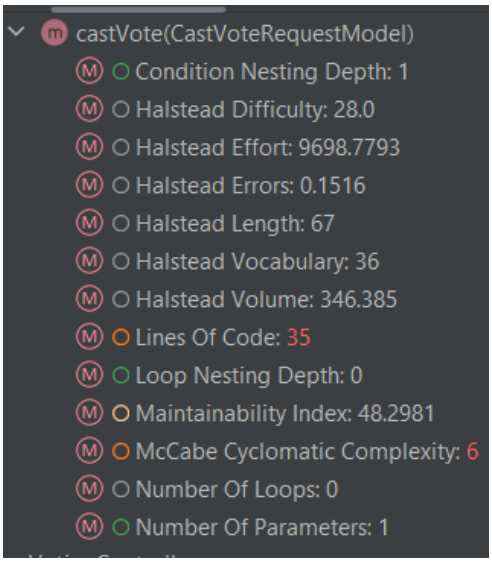
According to Filó, T. G., Bigonha, M., & Ferreira, K. 2015. A catalogue of thresholds for object-oriented software metrics. Proc. of the 1st SOFTENG, 48–55. The desired metric is around 4.

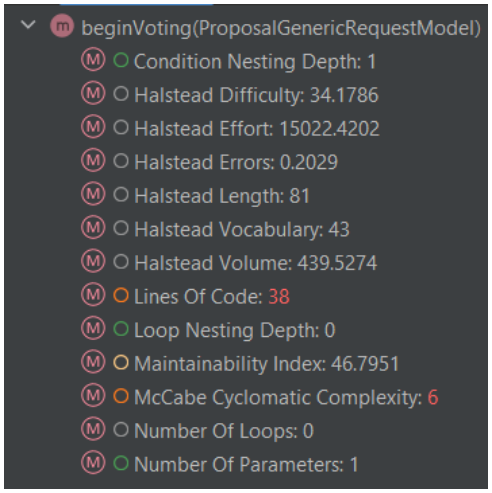
The MetricTree plugin was used to identify 5 methods which needed refactoring, namely











As you can see by the materials provided these 5 methods all go over the metric set for acceptable Lines of code (30) which is a method level code smell.

Task 2

Class level metrics

We split the **VotingController** class into three separate classes as it was quite bloated with long and complex methods. This drastically improved many metrics, namely: **ATFD**, **CBO**, **NOM**, **RFC**, **WMC**.

Before:

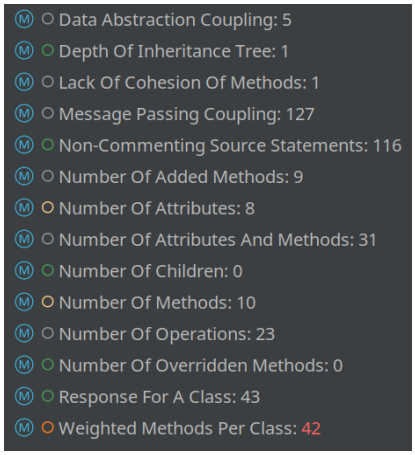
Class	ATFD	CBO	DIT	NCSS	NOAC	NOA	NOC	NOM	NOOM	NOPA	RFC	TCC	WOC	WMC
VotingController	10	30	1	146	0	2	0	9	0	0	83	0.75	1.0	39

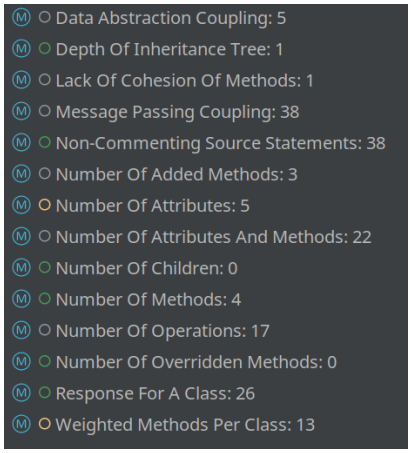
After:

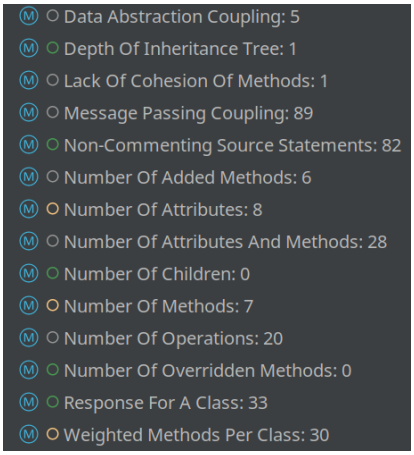
Class	ATFD	CBO	DIT	NCSS	NOAC	NOA	NOC	NOM	NOOM	NOPA	RFC	TCC	WOC	WMC
VotingManagerController	4	17	1	54	0	2	0	2	0	0	33	0.0	1.0	14
VotingInteractionController	5	11	1	39	0	2	0	4	0	0	39	0.3333	1.0	12
VotingInformationController	6	12	1	57	0	2	0	5	0	0	39	1.0	1.0	15

We refactored the **MemberController** class similarly, to improve the WMC metric, as it contained a lot of complex methods. The class was split into two, **MemberAttributeController** and **MemberInfoController**.

MemberController -> MemberAttributeController MemberInfoController

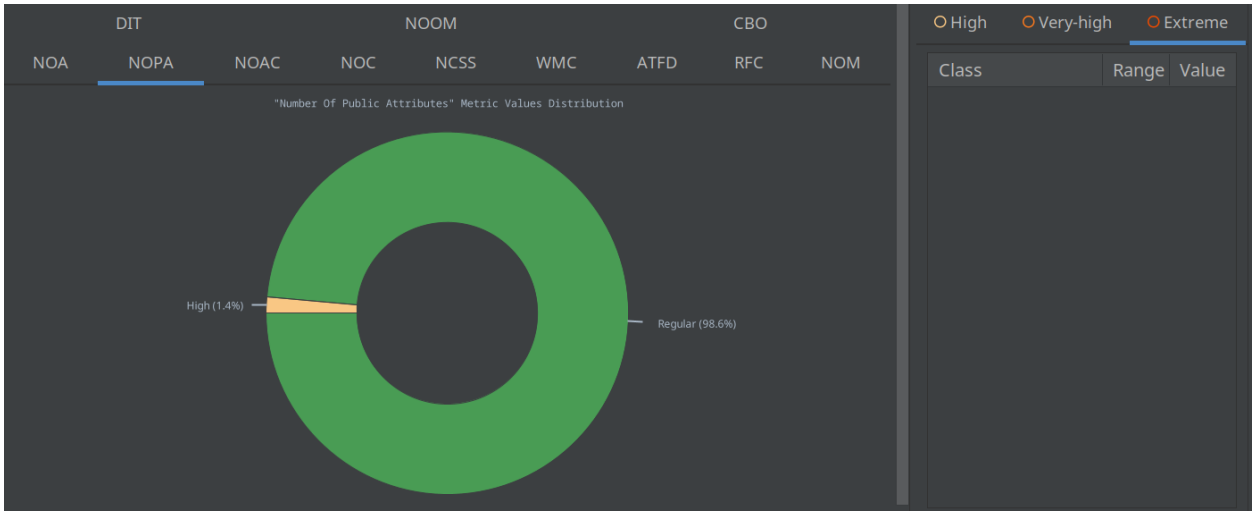






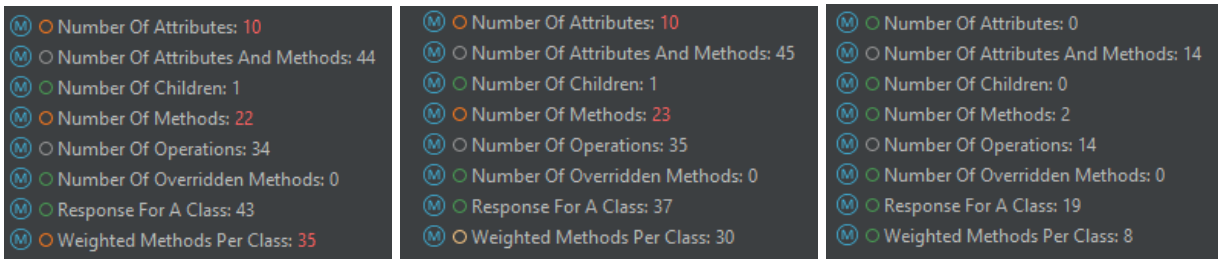
As pointed out earlier, the HoaModifyDTO class had both getters and setters, while having all its attributes set to public. We have changed all of them to private, and since many other methods

accessed these attributes via public access instead of getters and setters, we had to refactor many other methods in other classes to be using the getters and setters properly. After that, the NOPA metric for the HoaModifyDTO class decreased to 0 which is within the regular limits.



Similarly, to reduce the WMC metric of the **Proposal** class, some of its methods have been moved to a helper class, converting them to static functions. This allows the class to be more readable without hampering the functionality.

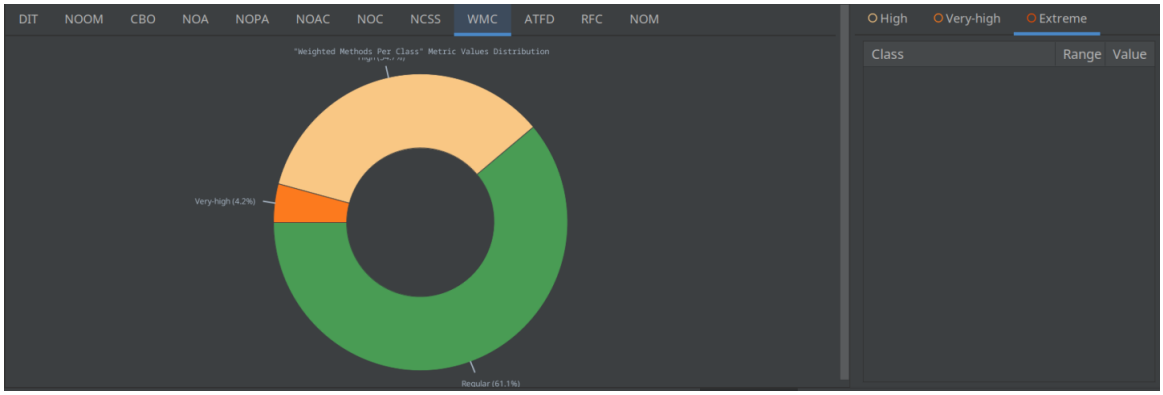
Proposal -> **Proposal** (updated) **ProposalHelper**



For the fifth class to refactor, we have not found anything else going over our thresholds. Instead, as per the assignment description, we would like to point to refactoring we did in previous weeks, specifically **commits 4fa4f52e972bb8111a3ef079d67789c039faf857 and 917d06f5cf9dd93bfb543ebcf4e6ed19eca03dc9**, also tagged on git as ‘refactoring’. In these commits, we spread out methods from AuthenticationController into UserInfoControllers, as the class was getting too long, and some new endpoints were created which were originally in AuthenticationController were also put into UserInfoController instead.

As we’ve been recently informed, lombok annotations should not be regarded as code smells, therefore we have replaced the below three methods with other refactoring above. We decided to leave this in the document nonetheless, but we do not count it towards the 5 classes refactored.

Refactoring for the three classes: **Membership**, **ProposalHistoryResponseModel** **ProposalInformationResponseModel** as per the **WMC** metric, results:



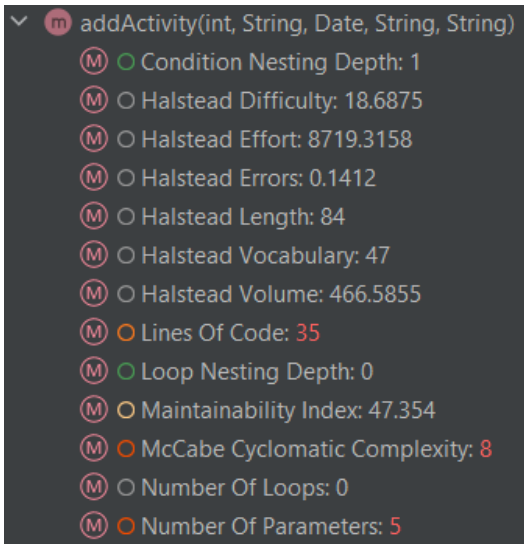
Upper row - before, Lower row - after

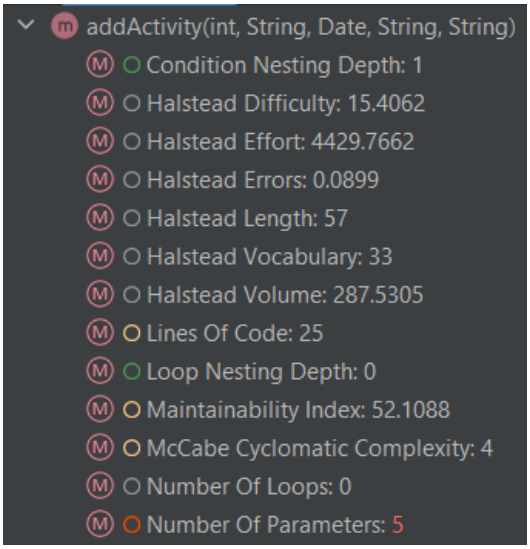
Membership	**ProposalHistoryResponseModel**	**ProposalInformationResponseModel**
Data Abstraction Coupling: 7	Data Abstraction Coupling: 5	Data Abstraction Coupling: 4
Depth Of Inheritance Tree: 1	Depth Of Inheritance Tree: 1	Depth Of Inheritance Tree: 1
Lack Of Cohesion Of Methods: 5	Lack Of Cohesion Of Methods: 9	Lack Of Cohesion Of Methods: 8
Message Passing Coupling: 26	Message Passing Coupling: 12	Message Passing Coupling: 13
Non-Commenting Source Statements: 51	Non-Commenting Source Statements: 12	Non-Commenting Source Statements: 13
Number Of Added Methods: 21	Number Of Added Methods: 17	Number Of Added Methods: 18
Number Of Attributes: 10	Number Of Attributes: 8	Number Of Attributes: 8
Number Of Attributes And Methods: 45	Number Of Attributes And Methods: 42	Number Of Attributes And Methods: 43
Number Of Children: 1	Number Of Children: 0	Number Of Children: 0
Number Of Methods: 22	Number Of Methods: 21	Number Of Methods: 22
Number Of Operations: 35	Number Of Operations: 34	Number Of Operations: 35
Number Of Overridden Methods: 0	Number Of Overridden Methods: 3	Number Of Overridden Methods: 3
Response For A Class: 43	Response For A Class: 33	Response For A Class: 34
Weighted Methods Per Class: 35	Weighted Methods Per Class: 45	Weighted Methods Per Class: 46
Data Abstraction Coupling: 2	Data Abstraction Coupling: 5	Data Abstraction Coupling: 4
Depth Of Inheritance Tree: 1	Depth Of Inheritance Tree: 1	Depth Of Inheritance Tree: 1
Lack Of Cohesion Of Methods: 8	Lack Of Cohesion Of Methods: 8	Lack Of Cohesion Of Methods: 7
Message Passing Coupling: 4	Message Passing Coupling: 19	Message Passing Coupling: 21
Non-Commenting Source Statements: 9	Non-Commenting Source Statements: 21	Non-Commenting Source Statements: 22
Number Of Added Methods: 10	Number Of Added Methods: 16	Number Of Added Methods: 17
Number Of Attributes: 10	Number Of Attributes: 8	Number Of Attributes: 8
Number Of Attributes And Methods: 37	Number Of Attributes And Methods: 40	Number Of Attributes And Methods: 41
Number Of Children: 0	Number Of Children: 0	Number Of Children: 0
Number Of Methods: 14	Number Of Methods: 19	Number Of Methods: 20
Number Of Operations: 27	Number Of Operations: 32	Number Of Operations: 33
Number Of Overridden Methods: 2	Number Of Overridden Methods: 2	Number Of Overridden Methods: 2
Response For A Class: 17	Response For A Class: 36	Response For A Class: 35
Weighted Methods Per Class: 18	Weighted Methods Per Class: 30	Weighted Methods Per Class: 31

The @Data annotation increased the cyclomatic complexity by a large amount and added a lot of extra methods for classes with a large amount of attributes. We have removed this annotation and replaced it with more specific annotations/implementations for methods used, thereby improving on the weighted methods per class metric, with minor improvements in a few other metrics as well for these three classes.

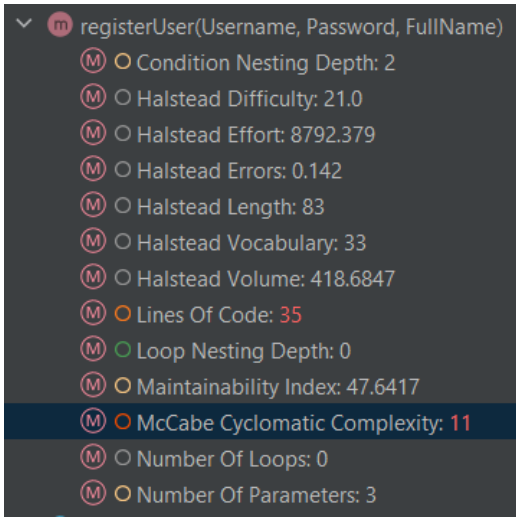
Method level metrics:

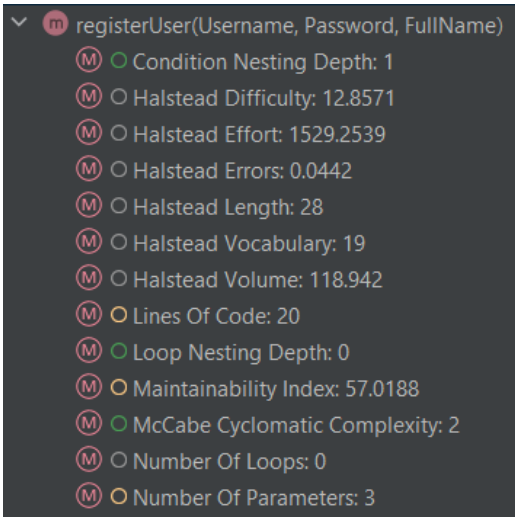
For the method addActivity() i moved all the logic that checks if the given variables are correct in size and by the specifications to a separate method that used to only check the name, that method was then changed form checkNameValidity() to checkParamValidity() and resulted in addActivity() having a lower Cyclomatic complexity and lines of code without impacting the helper method too much



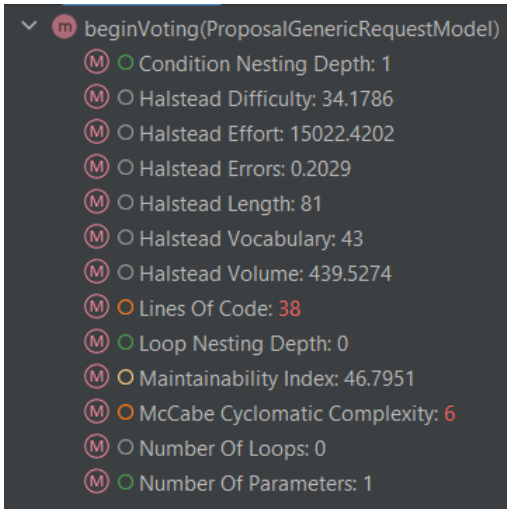


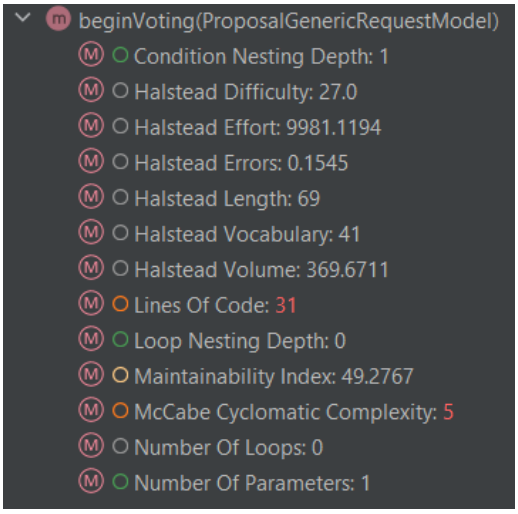
For the method registerUser() i once again moved the logic for checking the parameters to a separate method which resulted in a sizable difference in Cyclomatic complexity and Lines of code once again, both of which were bad smells over the limit



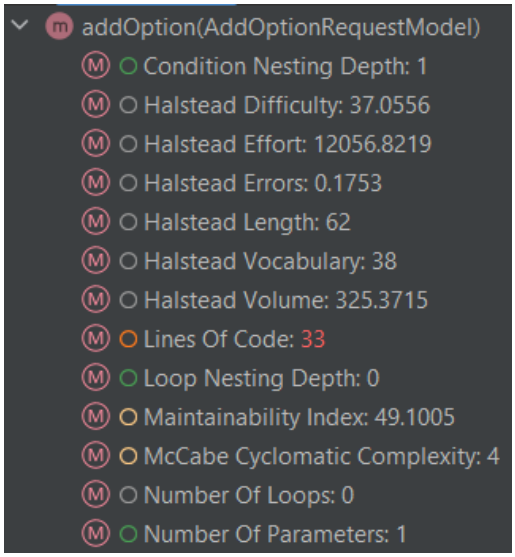


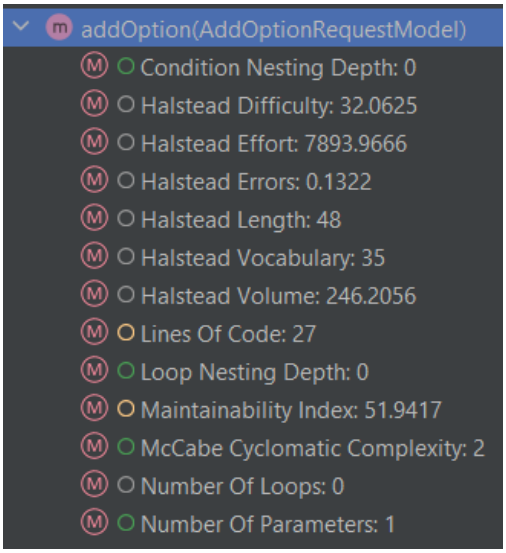
For the method beginVoting() for refactoring i removed some unnecessary checks, added the complex logic to the try/catch block as well as created constructors for a parameter which was set using setters which greatly reduced the Lines of code of the method





For the method addOption() for refactoring i once again removed unnecessary checks in favor of the try/catch block as well as moving the main logic inside the block as well





For the method castVote() I once again removed some unnecessary changes as well added the main logic to a try/catch block which decreased Lines of Code and Cyclomatic complexity once again