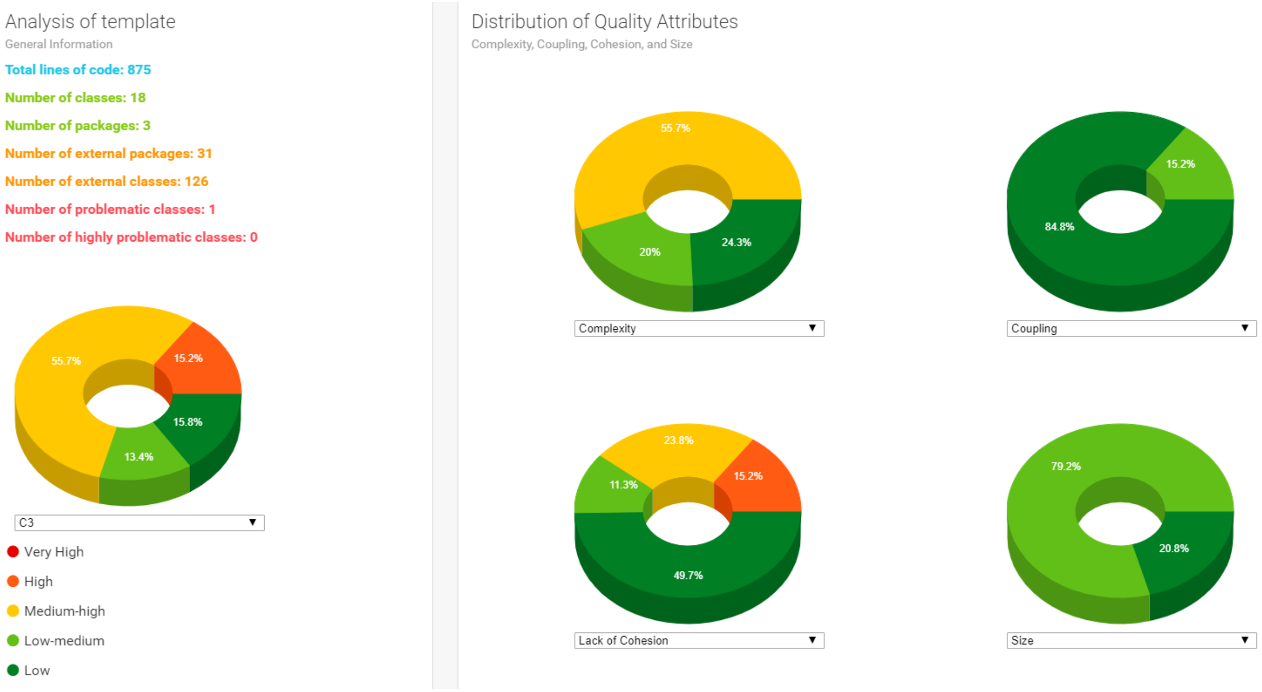
Refactoring

The tool we have used for computing the code metrics of our project is CodeMR which is a plugin in IntelliJ.



Analysis of template

General Information

Total lines of code: 875

Number of classes: 18

Number of packages: 3

Number of external packages: 31

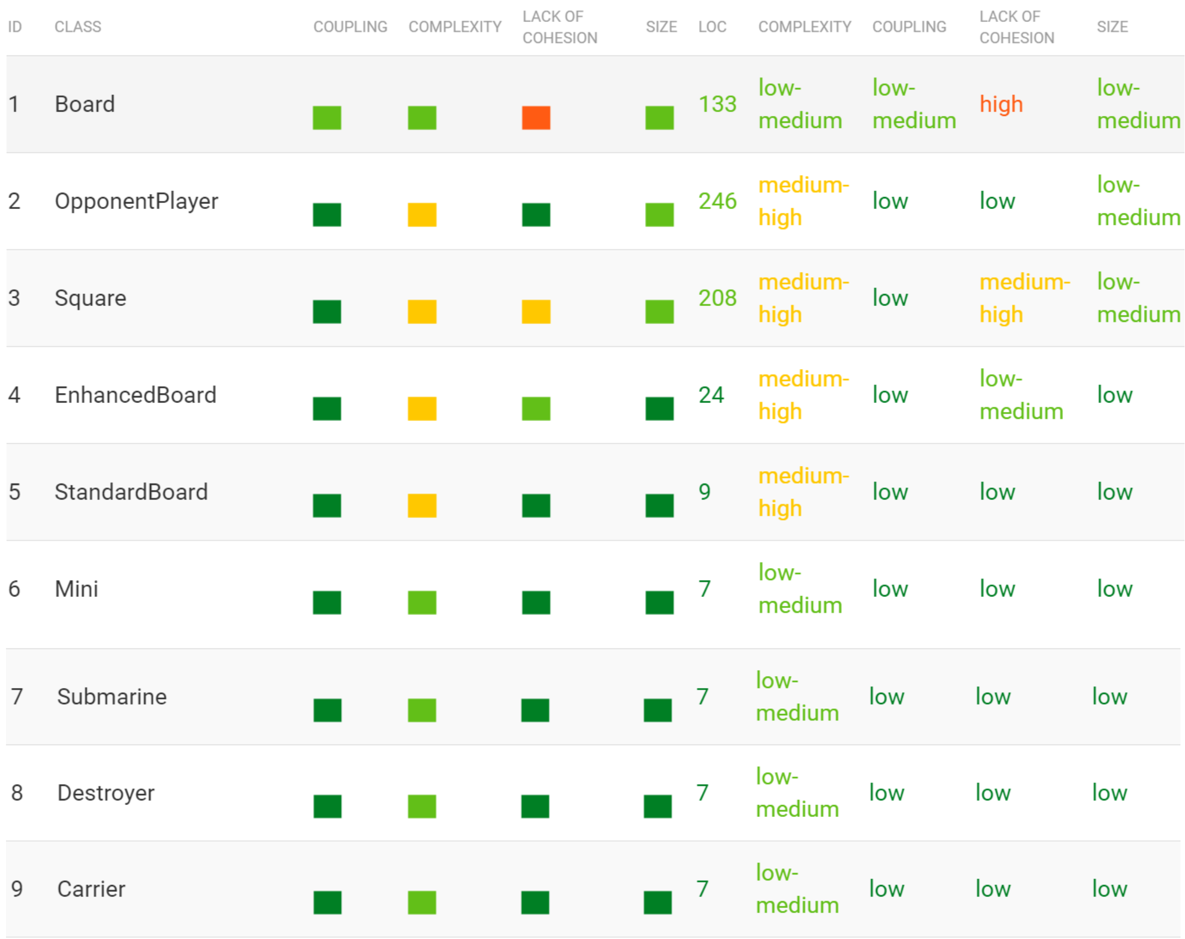
Number of external classes: 126

Number of problematic classes: 1

Number of highly problematic classes: 0

Distribution of Quality Attributes

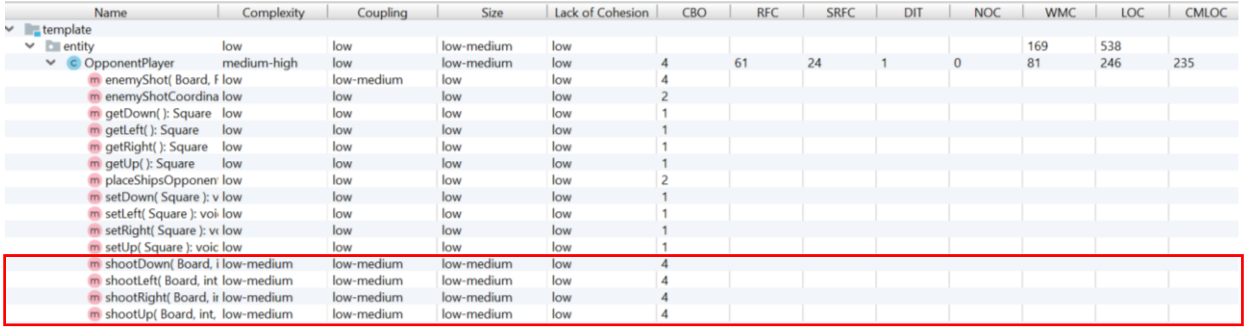
Complexity, Coupling, Cohesion, and Size



ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	Board	🟩	🟩	🟧	🟩	133	low-medium	low-medium	high	low-medium
2	OpponentPlayer	🟩	🟨	🟩	🟩	246	medium-high	low	low	low-medium
3	Square	🟩	🟨	🟨	🟩	208	medium-high	low	medium-high	low-medium
4	EnhancedBoard	🟩	🟨	🟩	🟩	24	medium-high	low	low-medium	low
5	StandardBoard	🟩	🟨	🟩	🟩	9	medium-high	low	low	low
6	Mini	🟩	🟩	🟩	🟩	7	low-medium	low	low	low
7	Submarine	🟩	🟩	🟩	🟩	7	low-medium	low	low	low
8	Destroyer	🟩	🟩	🟩	🟩	7	low-medium	low	low	low
9	Carrier	🟩	🟩	🟩	🟩	7	low-medium	low	low	low

10	BattleShip	■	▨	■	■	7	low-medium	low	low	low
11	Cruiser	■	▨	■	■	7	low-medium	low	low	low
12	StandardBoardCreator	■	■	■	▨	54	low	low	low	low-medium
13	EnhancedBoardCreator	■	■	■	▨	52	low	low	low	low-medium
14	Student	■	■	▨	■	29	low	low	low-medium	low
15	User	■	■	■	■	29	low	low	low	low
16	Game	■	■	▨	■	26	low	low	low-medium	low
17	Ship	■	■	▨	■	20	low	low	low-medium	low
18	BoardCreator	■	■	■	■	3	low	low	low	low

As to be seen the overall quality of our methods and classes is good. Firstly the Board class has to be refactored as indicated by the low cohesion, which indicates it is difficult to maintain, test, reuse, or even understand. Then also the OpponentPlayer, StandardBoard, EnhancedBoard class have to be refactored since their complexity is slightly high (medium). This increases the risk of unintentionally interfering with interactions and so increases the chance of introducing defects when making changes. Lastly Square class has to be refactored since the complexity and lack of cohesion are both high for the same reasons as mentioned above.



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC
template												
entity	low	low	low-medium	low						169	538	
OpponentPlayer	medium-high	low	low-medium	low	4	61	24	1	0	81	246	235
enemyShot(Board, F	low	low-medium	low	low	4							
enemyShotCoordina	low	low	low	low	2							
getDown( ): Square	low	low	low	low	1							
getLeft( ): Square	low	low	low	low	1							
getRight( ): Square	low	low	low	low	1							
getUp( ): Square	low	low	low	low	1							
placeShipsOpponen	low	low	low	low	2							
setDown(Square): v	low	low	low	low	1							
setLeft(Square): voi	low	low	low	low	1							
setRight(Square): v	low	low	low	low	1							
setUp(Square): voic	low	low	low	low	1							
shootDown(Board, i	low-medium	low-medium	low-medium	low	4							
shootLeft(Board, int	low-medium	low-medium	low-medium	low	4							
shootRight(Board, i	low-medium	low-medium	low-medium	low	4							
shootUp(Board, int,	low-medium	low-medium	low-medium	low	4							

The methods we have chosen to improve:

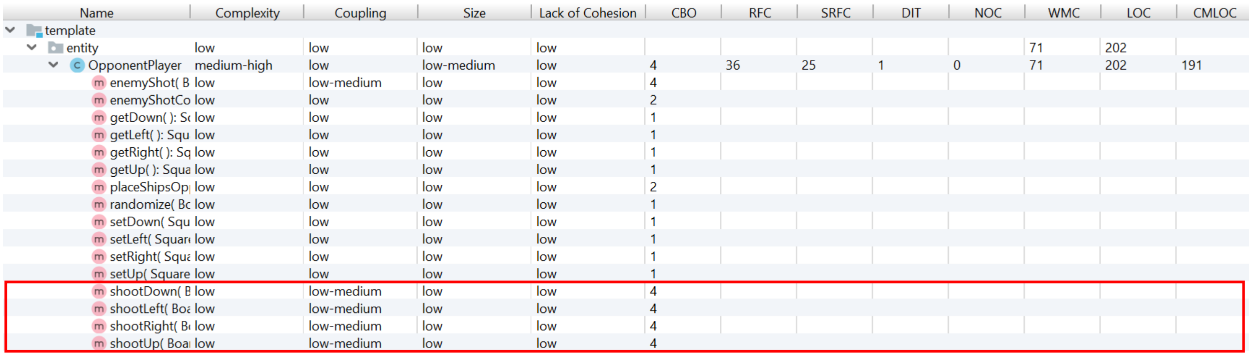
- OpponentPlayer.shootLeft()

- OpponentPlayer.shootRight()

- OpponentPlayer.shootUp()

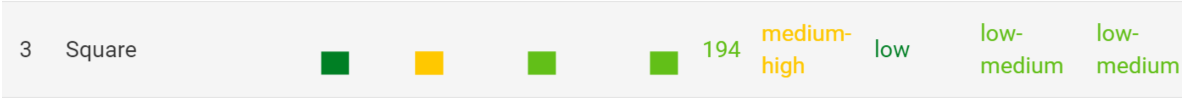
- OpponentPlayer.shootDown()

To improve these methods we have created a helpermethod named randomizedMove which is called by these methods instead of deciding a randomized move in each of them. This reduced the complexity of the methods. The coupling could not be reduced since this is caused by the fact that the method recursively calls itself when a ship in that certain direction is found. This is done for the intelligence of the computer. Removing this will make the quality of the code better, but will reduce the quality of the game drastically.



Name	Complexity	Coupling	Size	Lack of Cohesion	CBO	RFC	SRFC	DIT	NOC	WMC	LOC	CMLOC
template												
entity	low	low	low	low						71	202	
OpponentPlayer	medium-high	low	low-medium	low	4	36	25	1	0	71	202	191
enemyShot(B	low	low-medium	low	low	4							
enemyShotCo	low	low	low	low	2							
getDown( ): S	low	low	low	low	1							
getLeft( ): Squ	low	low	low	low	1							
getRight( ): Sq	low	low	low	low	1							
getUp( ): Squa	low	low	low	low	1							
placeShipsOp	low	low	low	low	2							
randomize(Bc	low	low	low	low	1							
setDown(Squ	low	low	low	low	1							
setLeft(Squar	low	low	low	low	1							
setRight(Squa	low	low	low	low	1							
setUp(Square	low	low	low	low	1							
shootDown(B	low	low-medium	low	low	4							
shootLeft(Boa	low	low-medium	low	low	4							
shootRight(B	low	low-medium	low	low	4							
shootUp(Boa	low	low-medium	low	low	4							

To increase the cohesion of the square class we have removed methods such as getters and setters that weren't used. Consequently these variables are declared as private transient.

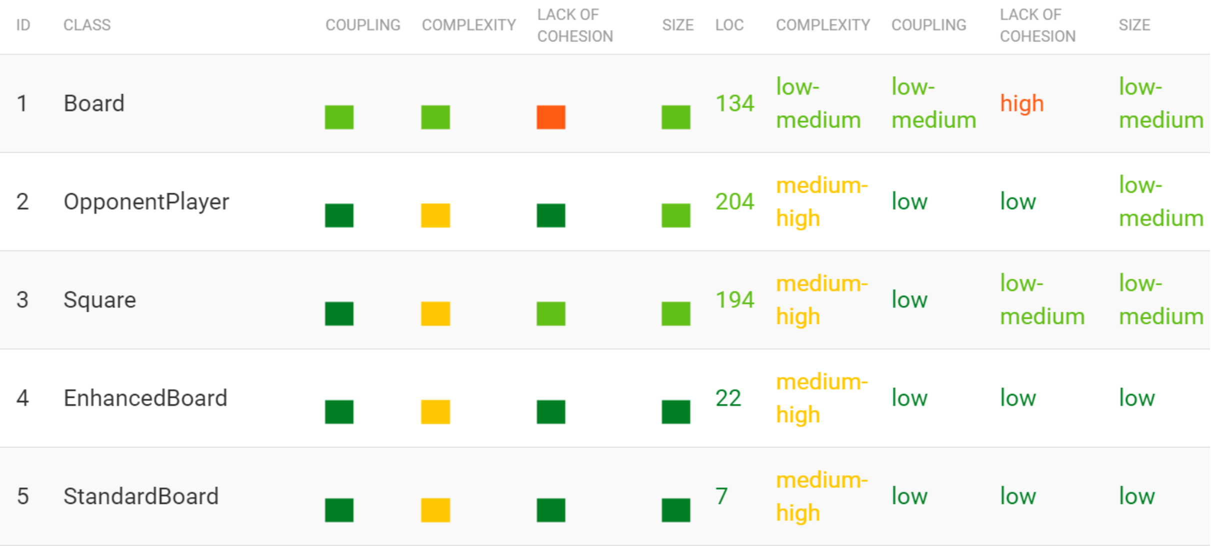


3	Square	🟩	🟨	🟩	🟩	194	medium-high	low	low-medium	low-medium

The complexity of the of the OpponentPlayer class cannot be improved the moment since this is caused by the fact that for the intelligence of the computer we use recursion. Removing this will make the quality of the class better, but will reduce the quality of the game drastically.

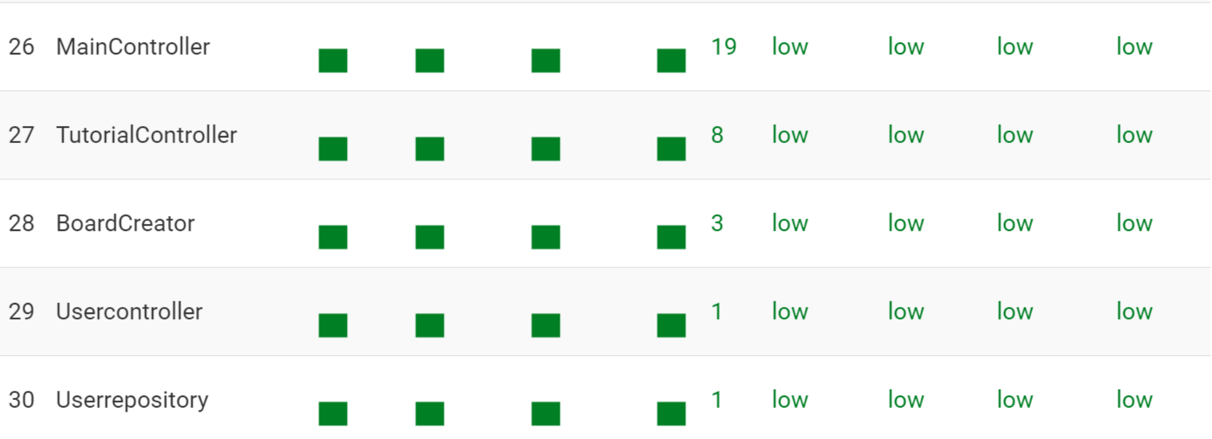
The complexity of the StandardBoard and EnhancedBoard can also not be improved at the moment since this is caused by applying the factory method design pattern to the board. The code seems at the moment unnecessarily complex, but when we want to adapt the boards for future releases this structure will make it very easy.

The complexity of the Square can also not be improved at the moment since this is caused by the fact that for marking the ships when they are entirely found we use recursion. This the mean feature which makes our game unique. Removing this would improve the quality of the code, but causes our game to lose its uniqueness.

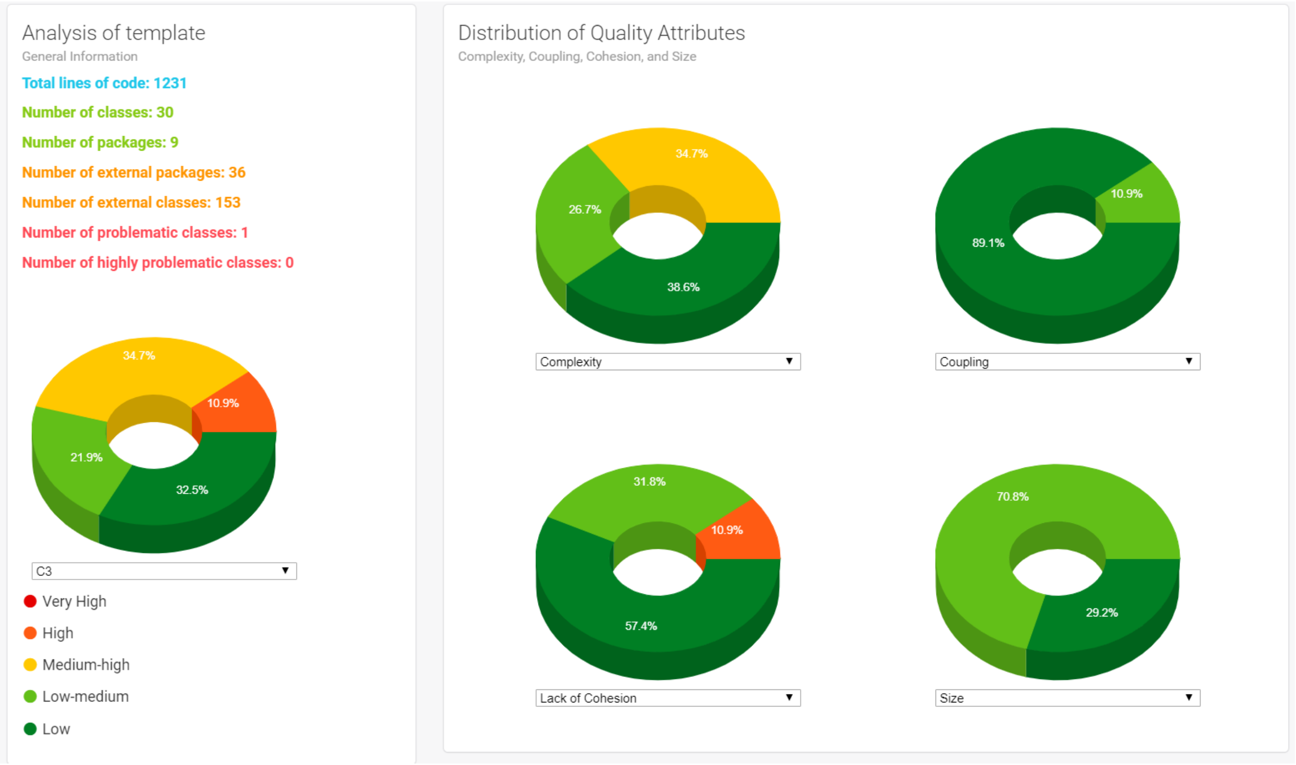


ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	Board	🟩	🟩	🟧	🟩	134	low-medium	low-medium	high	low-medium
2	OpponentPlayer	🟩	🟨	🟩	🟩	204	medium-high	low	low	low-medium
3	Square	🟩	🟨	🟩	🟩	194	medium-high	low	low-medium	low-medium
4	EnhancedBoard	🟩	🟨	🟩	🟩	22	medium-high	low	low	low
5	StandardBoard	🟩	🟨	🟩	🟩	7	medium-high	low	low	low

6	Scoring	🟩	▨	▨	▨	122	low-medium	low	low-medium	low-medium
7	Main	🟩	▨	🟩	🟩	20	low-medium	low	low	low
8	HelloWorld	🟩	▨	🟩	🟩	11	low-medium	low	low	low
9	Mini	🟩	▨	🟩	🟩	7	low-medium	low	low	low
10	Submarine	🟩	▨	🟩	🟩	7	low-medium	low	low	low
11	Destroyer	🟩	▨	🟩	🟩	7	low-medium	low	low	low
12	Carrier	🟩	▨	🟩	🟩	7	low-medium	low	low	low
13	BattleShip	🟩	▨	🟩	🟩	7	low-medium	low	low	low
14	Cruiser	🟩	▨	🟩	🟩	7	low-medium	low	low	low
15	Connect	🟩	🟩	🟩	▨	112	low	low	low	low-medium
16	StandardBoardCreator	🟩	🟩	🟩	▨	54	low	low	low	low-medium
17	EnhancedBoardCreator	🟩	🟩	🟩	▨	52	low	low	low	low-medium
18	HomePageController	🟩	🟩	🟩	🟩	34	low	low	low	low
19	RegisterController	🟩	🟩	🟩	🟩	31	low	low	low	low
20	LoginController	🟩	🟩	🟩	🟩	31	low	low	low	low
21	Student	🟩	🟩	▨	🟩	29	low	low	low-medium	low
22	User	🟩	🟩	🟩	🟩	29	low	low	low	low
23	Game	🟩	🟩	▨	🟩	26	low	low	low-medium	low
24	LeaderboardContro...	🟩	🟩	🟩	🟩	25	low	low	low	low
25	Ship	🟩	🟩	▨	🟩	20	low	low	low-medium	low



26	MainController	🟩	🟩	🟩	🟩	19	low	low	low	low
27	TutorialController	🟩	🟩	🟩	🟩	8	low	low	low	low
28	BoardCreator	🟩	🟩	🟩	🟩	3	low	low	low	low
29	Usercontroller	🟩	🟩	🟩	🟩	1	low	low	low	low
30	Userrepository	🟩	🟩	🟩	🟩	1	low	low	low	low



Analysis of template

General Information

Total lines of code: 1231

Number of classes: 30

Number of packages: 9

Number of external packages: 36

Number of external classes: 153

Number of problematic classes: 1

Number of highly problematic classes: 0

Distribution of Quality Attributes

Complexity, Coupling, Cohesion, and Size

We didn’t succeed in increasing the cohesion of the Board class. The reason of lack of cohesion is unknown.