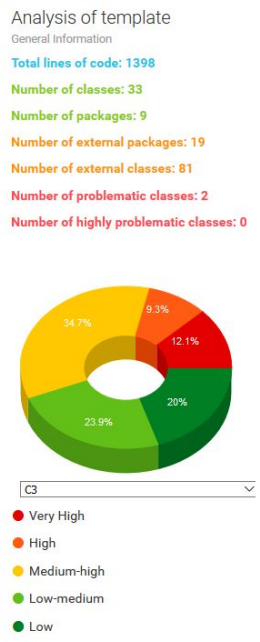
Refactoring report

Group 45

Before refactoring

Before refactoring , we made a screenshot of our CodeMR report. We had 2 problematic classes. GameScreen and GameOperator. Both had issues with coupling and lack of cohesion. We have managed to fix this after refactoring.



List of all classes (#33)

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION
1	GameScreen					169	low-medium	**very-high**	high
2	GameOperator					130	low-medium	medium-high	high
3	LoginScreen					30	low-medium	medium-high	low
4	SignupScreen					30	low-medium	medium-high	low
5	PreGameScreen					94	low	medium-high	medium-high
6	HowToPlayScreen					73	low	medium-high	medium-high

Class-Level refactoring

For refactoring of classes, we mostly used the extraction method. Underneath is a table of what is extracted. We first applied the template design pattern to the screens, making a screenbase and letting all other screens extend from it. Because of this we could split all the basic things needed for every screen into its separate base class. Because of the screen base class, all classes for all screens improved because less coupling occurred. Also its a fact that coupling occurs in the screen classes because we use a lot of LibGDX classes, so this can only be minimized to a certain extent. In total we have improved 8 classes doing this. The biggest and most important changes are documented below in the table and screenshots.

FIELD/METHOD	**OLD CLASS**	**NEW CLASS**
Sound	GameScreen	ScreenBase
BackgroundTexture	GameScreen	ScreenBase
Game	GameScreen	ScreenBase
SpriteBatch	GameScreen	ScreenBase
Camera	GameScreen	ScreenBase
Stage	GameScreen	ScreenBase
renderHelper()	GameScreen	ScreenBase

Before:

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION
1	GameScreen					169	low-medium	**very-high**	high

After:

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION
1	GameScreen					97	low-medium	medium-high	medium-high

For the refactoring of the GameOperator we did a similar thing, the only difference is that the extraction of the GameOperator class was more about extracting methods than extracting fields.

FIELD/METHOD	**NEW CLASS**
createBody()	BodyGenerator
makePitch()	BodyGenerator
makePaddle()	BodyGenerator
makePuck()	BodyGenerator
createFixtureDef()	BodyGenerator

Before:

2	GameOperator					130	low-medium	medium-high	high

After:

ID	CLASS	COUPLING	COMPLEXITY	LACK OF COHESION	SIZE	LOC	COMPLEXITY	COUPLING	LACK OF COHESION	SIZE
1	GameOperator					37	low	medium-high	medium-high	low

Method-level refactoring

com.mygdx.airhockey.movement.AiMovementController.updateVelocity

Before

class	OCavg	WMC
com.mygdx.airhockey.movement.AiMovementControlle	3,33	10

After

method	ev(G)	iv(G)	v(G)
com.mygdx.airhockey.movement.AiMovementControlle	1	3	3
com.mygdx.airhockey.movement.AiMovementControlle	1	3	3
com.mygdx.airhockey.movement.AiMovementControlle	1	2	2
com.mygdx.airhockey.movement.AiMovementControlle	1	1	1
com.mygdx.airhockey.movement.AiMovementControlle	1	4	4
Total	**6**	**14**	**14**
Average	1,00	2,33	2,33

Improved average cyclomatic complexity by splitting up method into two methods

com.mygdx.airhockey.database.tables.Score.equals

Before:

method	ev(G)	iv(G)	v(G)
com.mygdx.airhockey.database.tables.Score.setChosenN	1	1	1
com.mygdx.airhockey.database.tables.Score.setGameId(	1	1	1
com.mygdx.airhockey.database.tables.Score.setPoints(ir	1	1	1
com.mygdx.airhockey.database.tables.Score.setUsernar	1	1	1
com.mygdx.airhockey.database.tables.Score.toString()	1	1	1
Total	**14**	**16**	**18**
Average	1,17	1,33	1,50

After:

method	ev(G)	iv(G)	v(G)
com.mygdx.airhockey.database.tables.Score.setChosenN	1	1	1
com.mygdx.airhockey.database.tables.Score.setGameId(	1	1	1
com.mygdx.airhockey.database.tables.Score.setPoints(ir	1	1	1
com.mygdx.airhockey.database.tables.Score.setUsernar	1	1	1
com.mygdx.airhockey.database.tables.Score.toString()	1	1	1
Total	**15**	**17**	**19**
Average	1,15	1,31	1,46

Improved average cyclomatic complexity by splitting up method into two methods

Com.mygdx.airhockey.movement.KeyboardController.updateVelocity

Before:

method	ev(G)	iv(G)	v(G)
com.mygdx.airhockey.movement.KeyboardController.ge	1	1	1
com.mygdx.airhockey.movement.KeyboardController.Ke	1	1	1
com.mygdx.airhockey.movement.KeyboardController.se	1	1	1
com.mygdx.airhockey.movement.KeyboardController.se	1	1	1
com.mygdx.airhockey.movement.KeyboardController.up	1	3	7
Total	**6**	**8**	**12**
Average	1,00	1,33	2,00

After:

method	ev(G)	iv(G)	v(G)
com.mygdx.airhockey.movement.KeyboardController.Ke	1	1	1
com.mygdx.airhockey.movement.KeyboardController.se	1	1	1
com.mygdx.airhockey.movement.KeyboardController.se	1	1	1
com.mygdx.airhockey.movement.KeyboardController.up	1	3	3
com.mygdx.airhockey.movement.KeyboardController.up	1	1	5
Total	**7**	**9**	**13**
Average	1,00	1,29	1,86

Improved average cyclomatic complexity by splitting up method into two methods

com.mygdx.airhockey.screens.GameScreen.drawPlanet

Before:

Total	**25**
Average	1,67

After:

Total	**31**
Average	2,07

Increased Javadoc according to Javadoc metric.

Com.mygdx.airhockey.backend.GameOperator.makePuck

Com.mygdx.airhockey.backend.GameOperator.makePaddle

Before:

Total	**47**
Average	1,96

After

Total	**59**
Average	2,46

Increased Javadoc according to Javadoc metric.

The method-level metric tools gave almost no errors (only for the updateVelocity method which we fixed quickly) and we found it extremely difficult to find methods that could use refactoring. This is partially because we already did some unintentional refactoring during the project itself. Nevertheless do we think that our code quality right now is of good quality.