Refactoring and testing

20-time
Group 66

This document describes the design using responsibility driven design and UML. Starting with the drafted requirements.

Functional requirements

We use the MoSCoW model.

Must have

- The refactoring must not change or remove features of the game
- The refactoring must make the code easier to read and understand
- All code must be checkstyle compliant
- Unused libraries and variables must be removed
- Repeated (copy-pasted) code must be rewritten to have less overlap
- The automated tests must have boundary checking for the functions

Should have

- The code should use variables from the config file
- Balls and cannons should be relative to the position of the main level
- In multiplayer, the extra players should copy their layout from the original player (for extensibility)
- In multiplayer, the score and shoot timeout should be kept for each individual player
- Commented code should be removed
- All classes should have an equivalent test class, because of parallel inheritance
- All functions/branches should have an equivalent test function
- There should be a testing document, detailing how the test suite is designed

Could have

- The code could have comments in larger functions
- TODO comments could be fixed and removed from the code
- Errors, mistakes or bugs discovered by tests could be fixed immediately
- A new class could be created, where GameScreen and SplitGameScreen inherit from

Won’t have

- The automated tests will not cover all of the code

Non-functional requirements

Technical design
The code should work the same after refactoring. However, the underlying code will be different.
For the test classes we will use parallel inheritance, which means there will be a test class for each regular class in the code.

CRC

Class	Responsibility	Collaborations
[Classname]Test	Test class for the class which has to test the functionality	[Classname]

UML
This is a simple schematic overview of a small part of the code