Report Assignment 4

by

Group 33

Teaching Assistant
B. Reijm

Group members
Wytze Elhorst
Thomas Kolenbrander
Steven Meijer
Bart van Oort
Ruben Vrolijk

Contents

Exercise 1 - Your wish is my command

Multiplayer design
The requirements can be found in the document 'Requirements document Multiplayer.pdf' on our Github in /doc/

The first new class is Multiplayerboard, this is a subclass of Board. It has the same responsibilities as Board, but, since it gets controlled by keyboard, also has to keep track of the current targeted cell and it must be able to move targets. It has the same collaborators as the Board class.

Multiplayerboard	
Superclass: Board	
Subclass: none	
Responsibilities: Fill the cells on board with gems upon creation Recognise chains and remove them Refill empty cells Make the cells fall down properly Keep track of the current targeted cell. Move the target to a different cell.	Collaborators: Gem Cell Direction Position

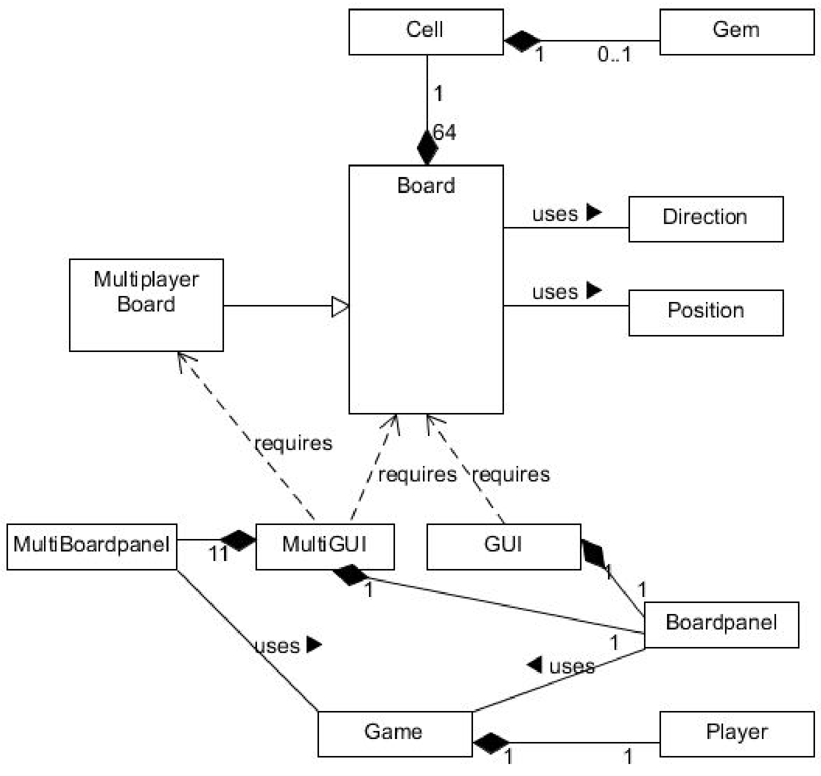
Another new class is the MultiBoardpanel, this class is responsible for creating a panel with a MultiBoardpanel on it and for giving a sprite to the target cell.

MultiBoardpanel	
Superclass: none	
Subclass: none	
Responsibilities: - Create a panel containing MultiPlayerboard. - Gives different sprites to cells that are targeted.	Collaborators: - GUI - Game

The third and final new class is MultiGUI, this class is responsible for creating the interface containing a Boardpanel and a MultiBoardpanel. Another responsibility is mapping all the buttons and actions for the second player to keys on the keyboard. It has the same collaborators as GUI with the addition of MultiBoardpanel.

MultiGUI	
Superclass: none	
Subclass: none	
Responsibilities: - Creates an interface containing a Boardpanel and at least one MultiBoardpanel. - Maps the buttons and actions for the second (and all aditional) players to keys on the keyboard. - Creates a panel containing the score of all players - Creates and maps buttons to restart, save and load the game.	Collaborators: - Board - MultiplayerBoard - Boardpanel - MultiBoardpanel

The class diagram for this design can be found below:



Fixing bugs in animations

In order to fix the bugs in our animations, we first made an overview of the bugs currently in the system. Each major bug was written down in its own issue on Github. When it was clear what went wrong, we started debugging. The debugging learned us that almost all faults happened because the positions of the ImageView-objects were not updated correctly. We fixed this by changing the property of the ImageView-objects that was being animated and consequently what the end values of these properties became. Once we had fixed this, most of our bugs disappeared, leaving only a few minor bugs, which were easily fixed by some small tweaks in the code.

Tests running on Travis CI

In order for Travis to run the tests, all tests should be located in ‘/test/java’. After we moved all the tests, Travis ran them just fine.

Exercise 2 - Software Metrics

2.1
The InCode analysis file is located in the /doc/ folder of our project.

2.2
Since our software does not contain any of the design flaws that inCode checks for, here a short note why such design flaws are not present in our system.

God class:
A god class or a god object could be found in the Board class, that we could have made so that the board knew everything and did all the work. However, we instead divided up all tasks to different classes, so one class only has one major task to complete. For example, the creation of the board is handled in a class separate from where the board is rendered. That is the reason there are no god classes in our system.

Data class:
A data class is also not present in our system, since we do not have to use constants in our implementation that would be useful to put in a data class. For this type of game, a data class would not be useful, so it is logical we don't have one.

Internal duplication
Internal duplication could in principle be found anywhere, but we have made sure that no code is duplicate and unnecessary. This not only reduces the amount of code our project has, but also helps keeping everything organized and accessible.