Report
Assignment 3

by

Group 33

Teaching Assistant
B. Reijm

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

Contents

Exercise 1 - 20-Time, Reloaded

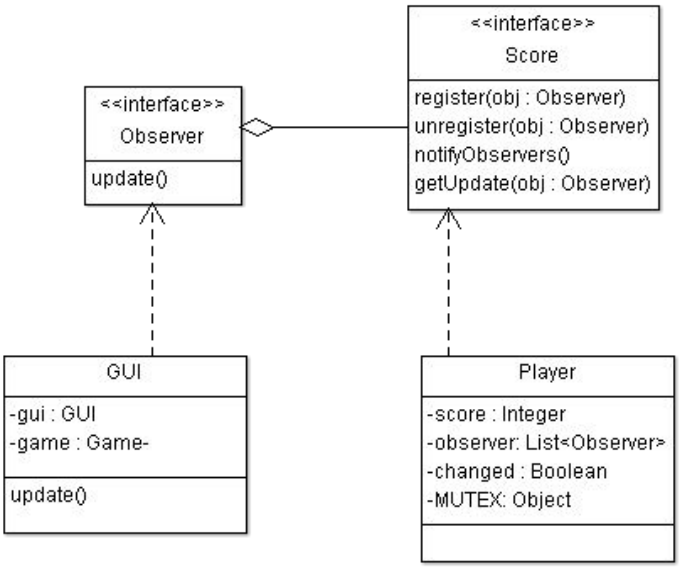
For this exercise we will implement animations in our game. We think animations are really important for the game experience. It makes it more clear what happens, and just makes the game look better overall.

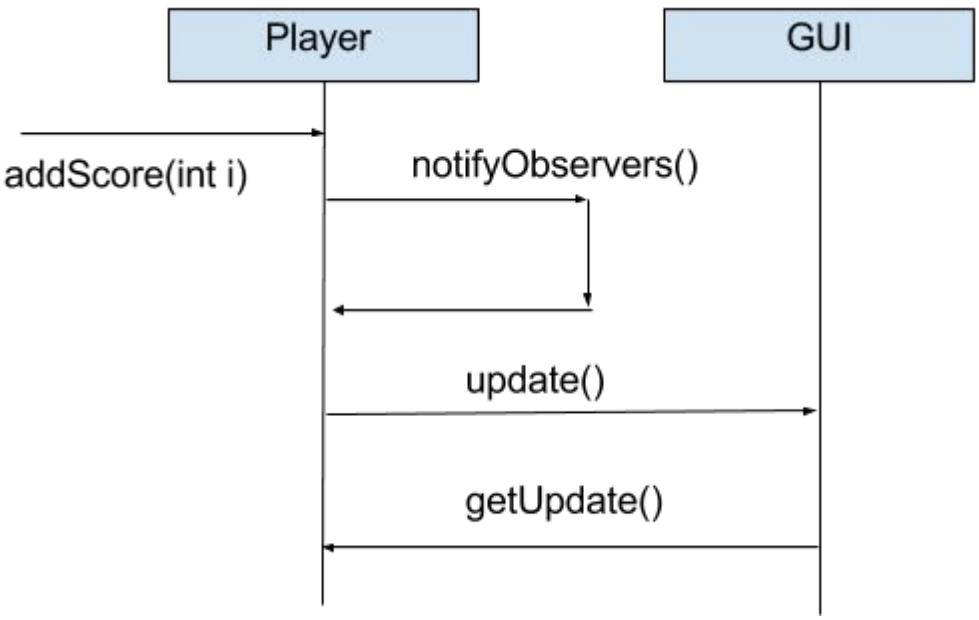
The way we will implement this is during the new generation of the board, all changes of the gems are saved in a list. We used a new change object for this with two positions, one with the initial position and one with the new position. This list with changes will be used to generate the animations. We couldn't figure out if the GUI or the game class should responsible for translating these changes into animations. In order to solve this problem we came up with a new class called the TimelineController. The TimelineController is responsible for generating the animations using the changes generated earlier. If we would not have done this, the GUI or the game class would have gotten too many responsibilities.

Exercise 2 - Design patterns

Observer pattern

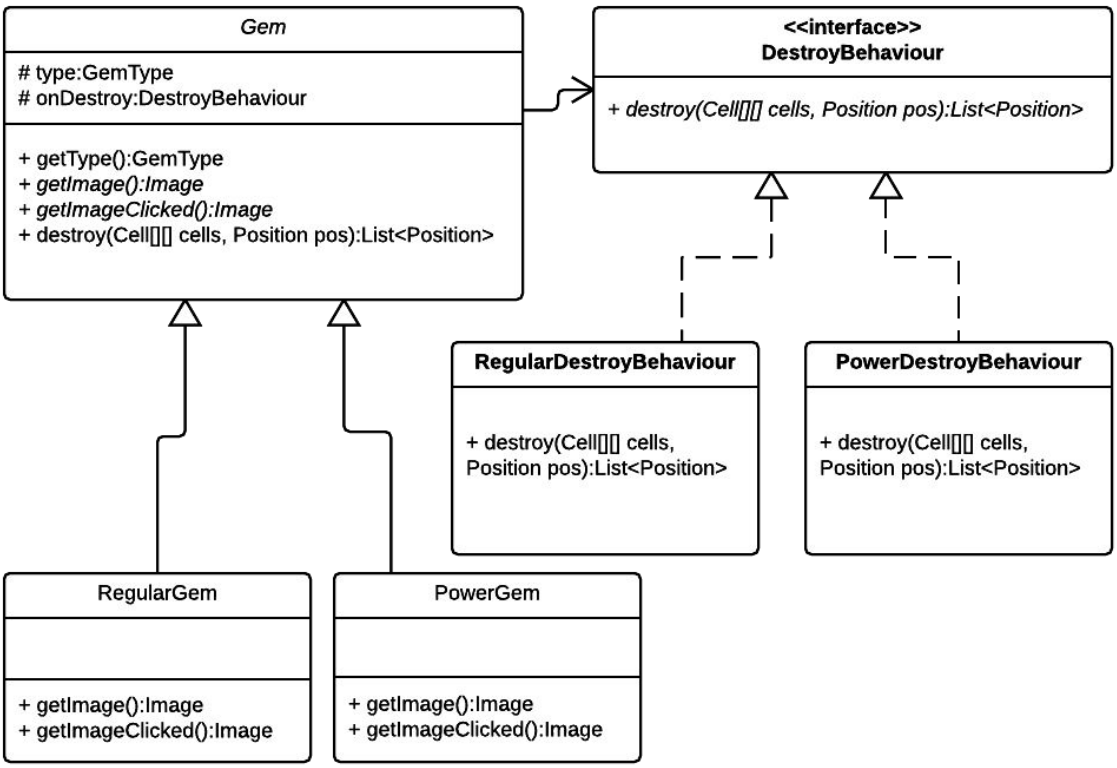
We have decided to implement the observer pattern to keep the score that the player as updated in the gui, without the need for the gui and the backend to communicate. This is good, as direct calls from the backend to the frontend decrease evolvability, and could cause small changes in the system to have large effects on the rest of the system, including to parts that should not be connected.

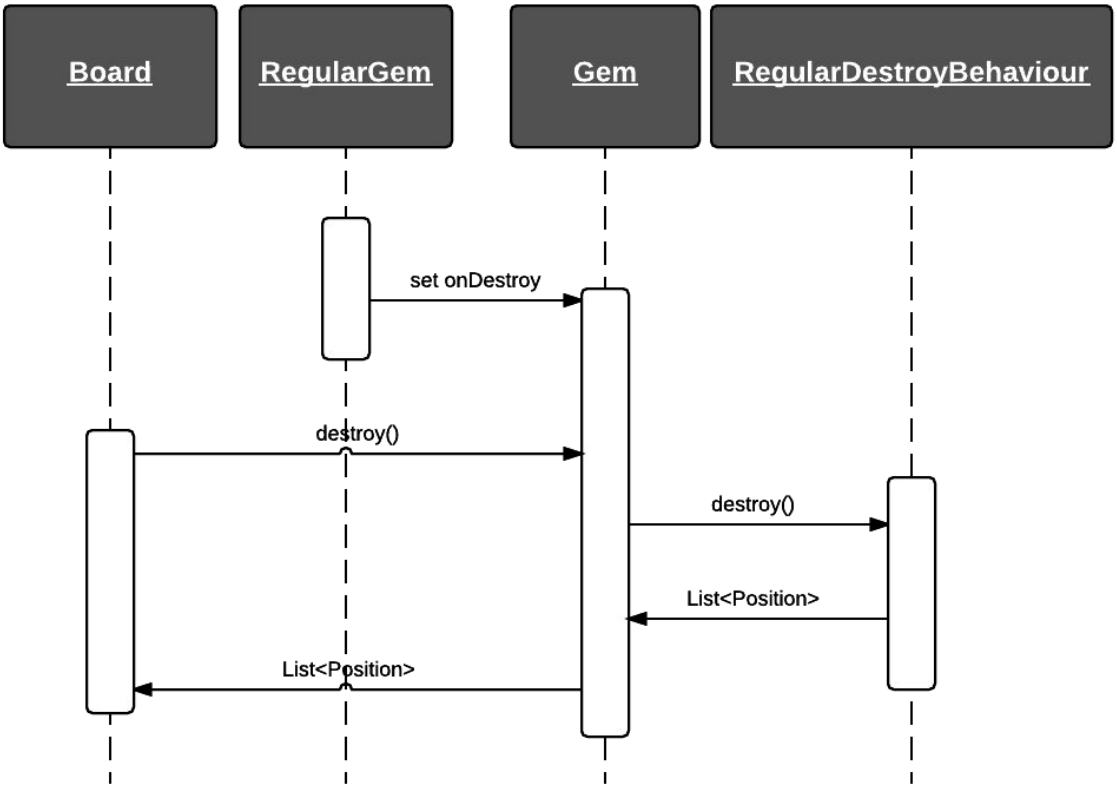




Strategy pattern

We chose to implement the strategy pattern for the different kinds of gems to make it easy to add new kinds of gems. This is used to easily implement the new PowerGems. The regular kind of gems are called RegularGems and Gem is now an abstract class. A gem now also has a DestroyBehaviour, which is called when a gem is destroyed. It handles the behaviour of what a gem does when it is destroyed, which is different for each different kind of gem. This is implemented by making an interface DestroyBehaviour and two concrete implementations for the different kind of gems.




Board
RegularGem
Gem
RegularDestroyBehaviour
set onDestroy
destroy()
destroy()
List<Position>
List<Position>

Exercise 3 - Software Engineering Economics

Exercise 3.1

In the article, they state that they relate success and failure to better and worse than the average performance of a project. So when comparing every single project to the average performance of all projects on the repository, then by analyzing the actions that were taking during the projects that performed better than average good practices can be found. Analyzing the actions taken during projects that performed below average can result into finding bad practises.

Exercise 3.2

Even though the Visual Basic was almost always a success factor, does not mean that it is necessarily the best programming language for successful good software projects. Since they had an upfront collection of project data and Visual basics was only used in 6 projects from 2 different companies, it can very well be that the people from those companies were just personally better at Visual Basic than at other languages.
Another possible explanation is that Visual Basic is less complex than a lot of other project environments and thus will end up as a good practise more often.
Since they didn't have the resources to confirm why it was that Visual Basic scored so well, it is not a very interesting find and would require further research before anything could really be concluded.

Exercise 3.3

1 - One thing they could have also researched was whether certain members of the team were only specialized in certain aspects of the project. This would mean the project is more dependent on the team working well together, so as long as the team has good teamwork this would probably end up as a good practise since every area of the project is being worked on by someone who specializes in it.

2 - They only researched the primarily used programming languages, so they could always add some more languages to be researched, say for instance Python. Even though the same problems could occur that occurred with Visual Basic, there is no harm in researching more languages. Since Python is also a relatively easy language, I'd expect it to do well and end up as a good practise.

3 - A final factor could be researching where the people worked on the project. Apart from all the meetings and the weekly sprints there has been no research on where the team actually works on the project. I think it would make a difference if someone was working at home or if the whole group was in a room working on it together. The former allows for less communication and would probably end up as a bad practise, while the latter allows for more cooperation and would thus end up as a good practise.

Exercise 3.4

1 - Many team changes, inexperienced team:
This is the most straight-forward one, if there are multiple changes in the team, it is going to take more time for everyone to adapt to this and keep up the teamwork, when the team is also inexperienced it becomes even more difficult to keep on schedule, since the team doesn't know how to handle setbacks like changes in the team well.

2 - Once-Only projects:
A project that only has to be done once and has probably not been done before. This means that the people working on it will have to learn new things which will only be used for this one project.This goes hand in hand with the above stating that inexperience is a bad practise.

3 - Dependencies with other systems:
This means that the software of the project depends on other software packages. This can create a lot of issues. The software can lose compatibility when the package it depends on has an update. When a lot of packages have dependencies with each other it can create a lot of those unnecessary problems. So even though it might seem handy to rely on other software packages, since that might seem faster, it can create more trouble than good and is thus a bad practise.