Documentation for the Breakout project

Team #42

The Breakout game is developed within the Java Capstone Series Pt. 1 course, held by openHPI in September, 2018, and is implemented in accordance with the given task.

2018

Table of contents

[UML diagram of the project 3](#_Toc525822172)

[Game description and gameplay 3](#_Toc525822173)

[High-level description 3](#_Toc525822174)

[Game elements and controls 4](#_Toc525822175)

[Screenshots 5](#_Toc525822176)

[Lab Report 6](#_Toc525822177)

[Appendix: 1](#_Toc525822178)4

[Current Version History Master Branch GIT 15](#_Toc525822179)

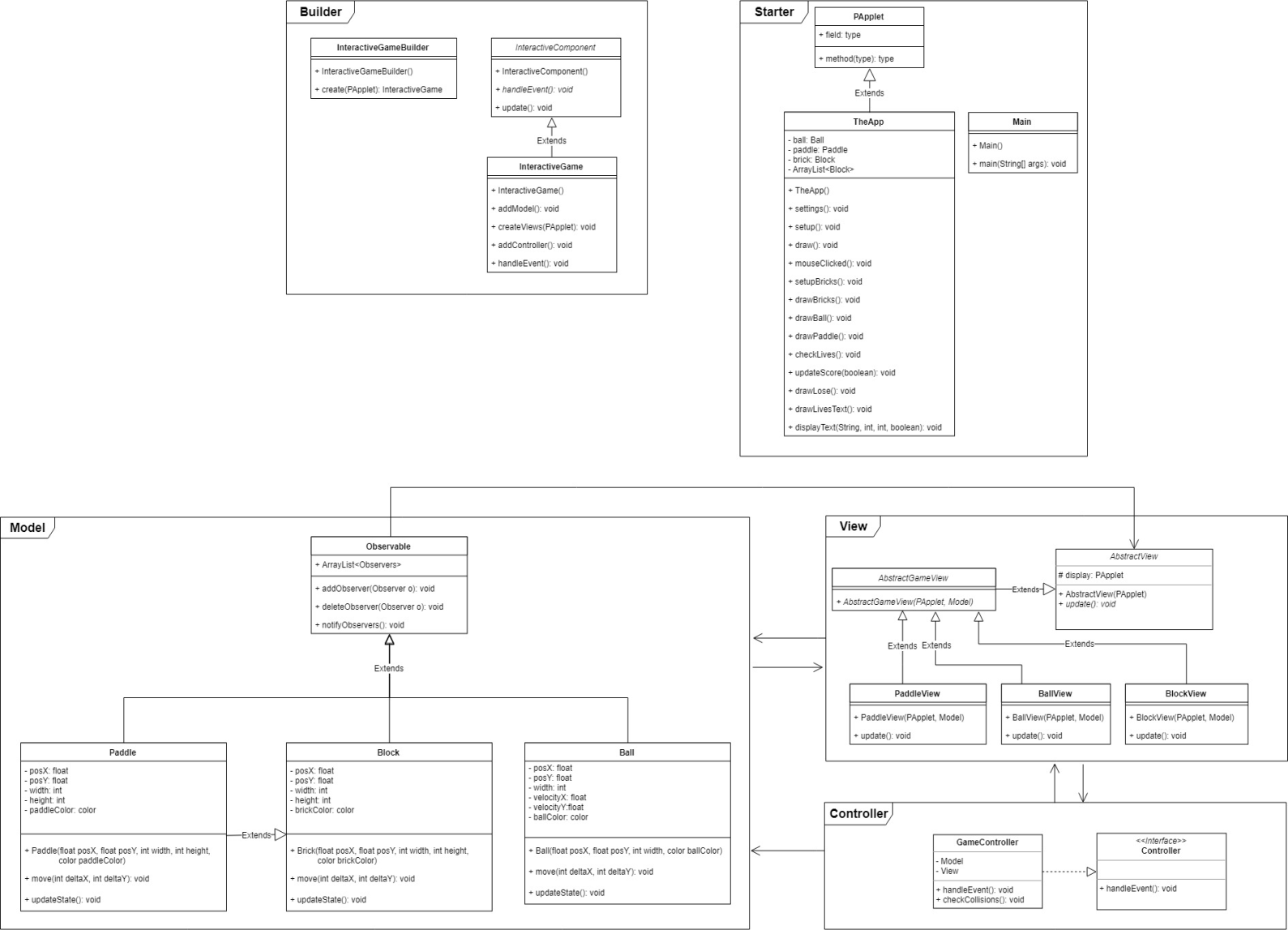
[Currents structure of classic part (waiting for conversion to MVC) 16](#_Toc525822180)

[Current MVC-Structure 17](#_Toc525822181)

[Current MVC-Structure 18](#_Toc525822182)

# UML diagram of the project

This is the UML class diagram of the project (picture 1). It shows the interfaces, abstract and concrete classes used in the project, as well as the relationships between them. The project was developed in accordance with the MVC architectural pattern.



Picture 1. UML class diagram.

# Game description and gameplay

## High-level description

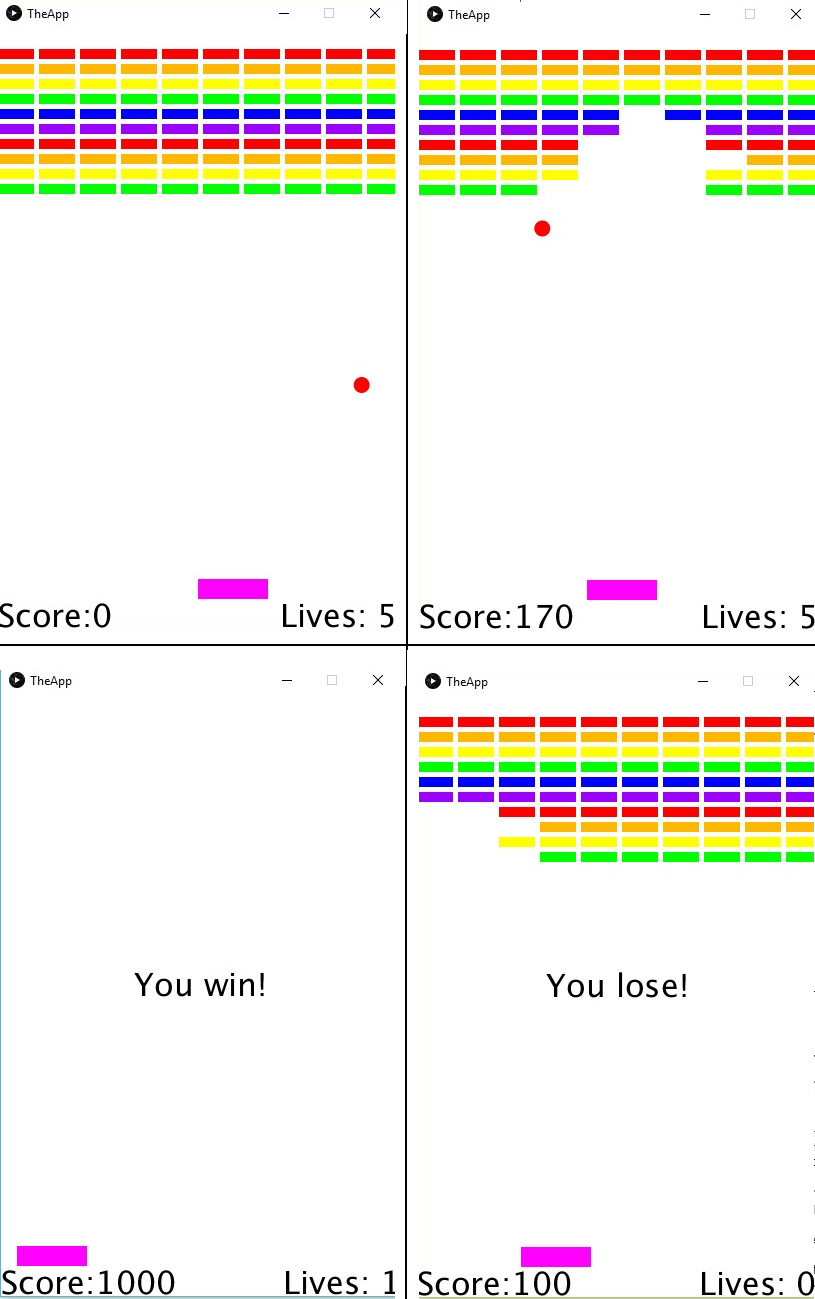
In *Breakout*, a layer of bricks lines the top third of the screen. A ball travels across the screen, bouncing off the top and side walls of the screen. When a brick is hit, the ball bounces away and the brick is destroyed. The player loses a turn when the ball touches the bottom of the screen. To prevent this from happening, the player has a movable paddle to bounce the ball upward, keeping it in play[[1]](#footnote-1).

## Game elements and controls

Table 1

|  |  |  |
| --- | --- | --- |
| Element | Description | Keys |
| Ball | Once inserted to the playground, it moves to a certain set of rules:   * Ball bounces (changes its direction) when it hits the left, right or upper bound of the playground. * Ball bounces when it hits the paddle. * Ball leaves the playground when it "hits" lower bound of the playground. * A new Ball is automatically inserted when the previous Ball has left the playground. | **⎯** |
| Paddle | The game element which is used to hit the ball. The Paddle can move to the right and to the left within the game window. The movement of the Paddle is controlled by the player. | Left arrow moves the Paddle to the left.  Right arrow moves the Paddle to the right. |
| Wall of Bricks | The Wall of Bricks contains 10 rows of 10 Bricks in each. When the player destroys all bricks, he wins and gets the message “You win!” | **⎯** |
| Lives | At the beginning of the game the player has 5 lives. Each time the Ball leaves the playground, the player loses 1 life. When the player loses all lives, he loses the game and gets the message “You lose!” | The number of lives decreases automatically to 0. |
| Score | The player receives 10 points for each destroyed Brick. The maximum number of points is 1000. Points are counted throughout the game until the player loses or wins. | The score counts automatically. |

## Screenshots

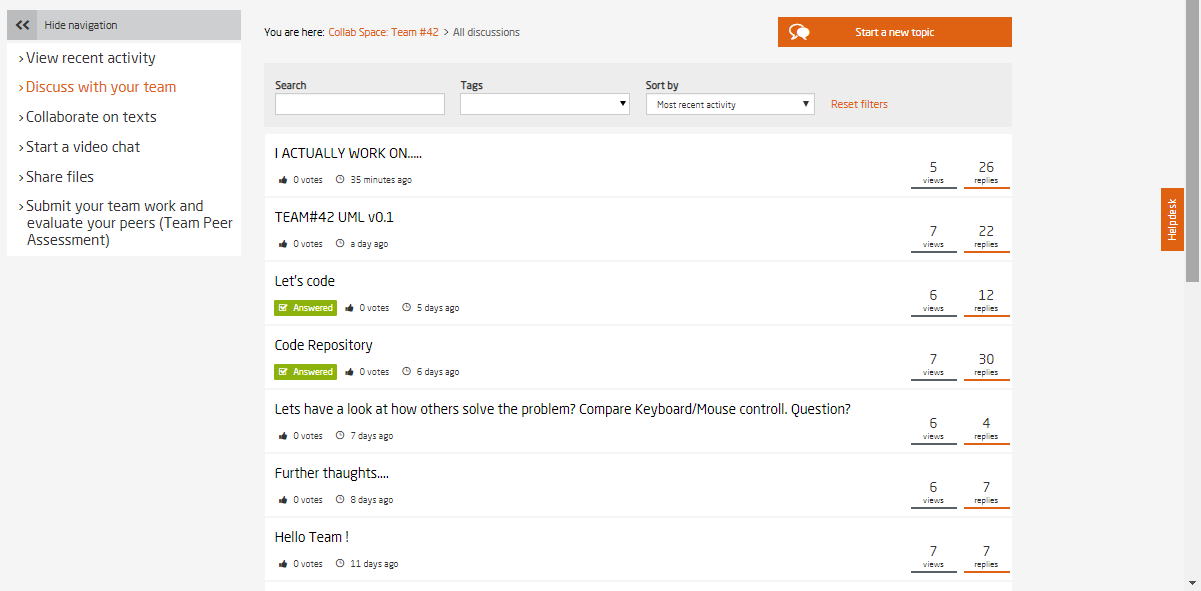


Picture 2. Screenshots of gameplay

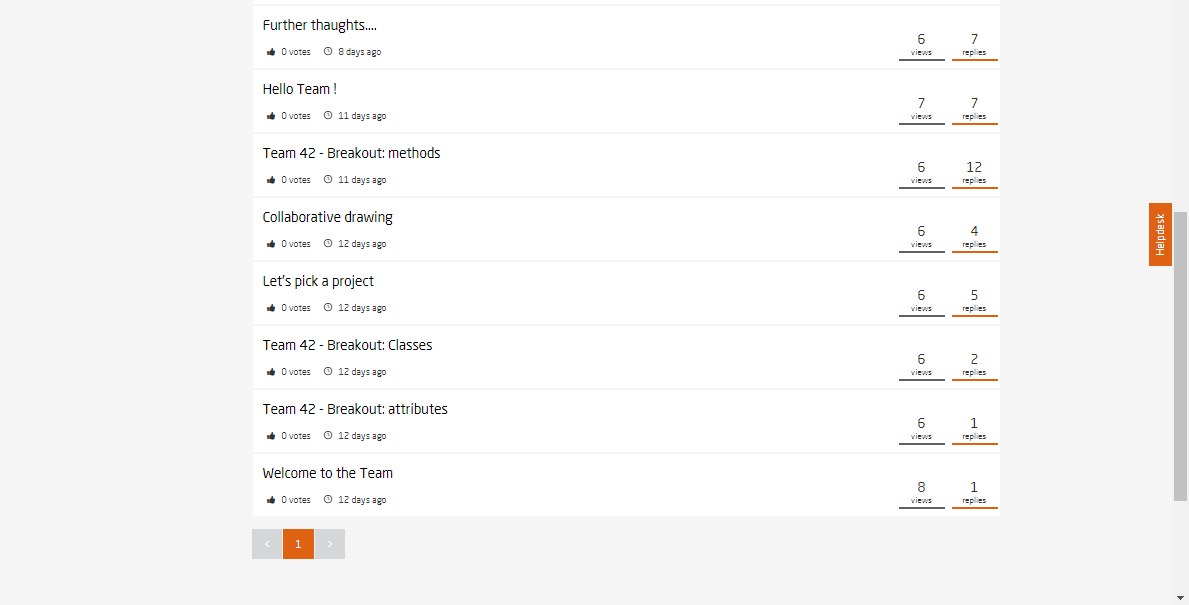
# Lab Report

In our team work began from the very first day. Almost immediately, 7 out of 8 team members got in touch, but only 3 eventually started to work directly on the project.

Throughout the course, we actively communicated at the forum in all 13 topics that we’ve created (pict. 3 - 4).



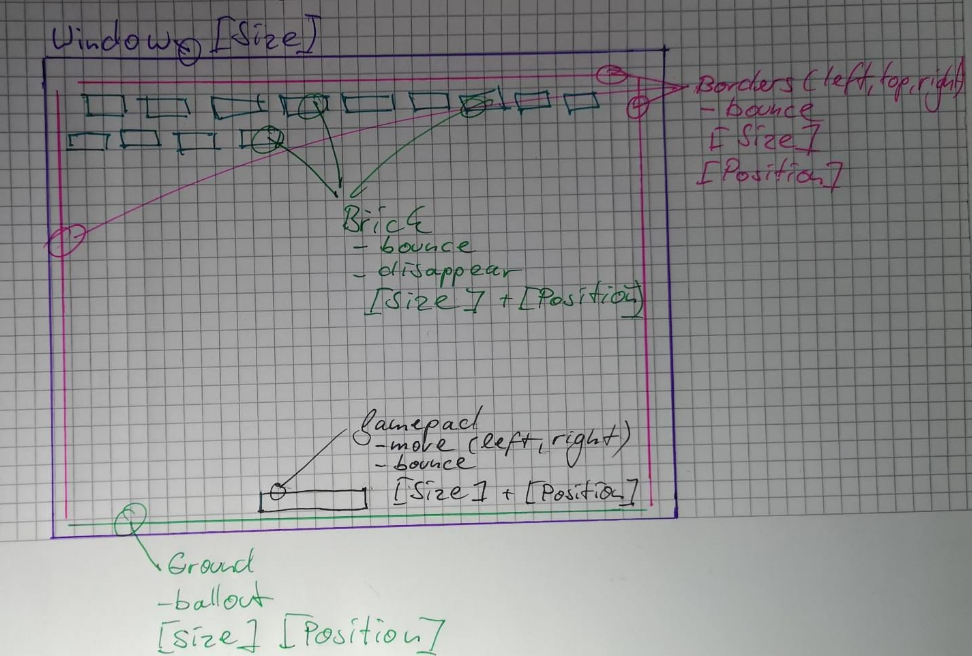
Picture 3. Team topics-1.



Picture 4. Team topics-2.

**During the initial discussions, the following was done:**

1. We selected a tool for drawing the UML class diagram, and it was [Draw.io](https://www.draw.io/).
2. We forked Tom's repository with the starter code.
3. We sketched out a list of necessary classes, attributes and methods.
4. We made a sketch of the game field (pict. 5).



Picture 5. Initial sketch of the game field.

We argued a lot about the ways of collision detection, as well as how to implement a Wall of Bricks. In this regard, there were 2 main positions:

1. create a 2d array [row, column] of Brick-Objects. The bricks (rect) themselves have attributes: x = start position x, y = startposition y, h = height, w = width.
2. create a 1d array where the two-dimensional location of the bricks is defined by the coordinates stored in the objects.

As a result, the first option was chosen.

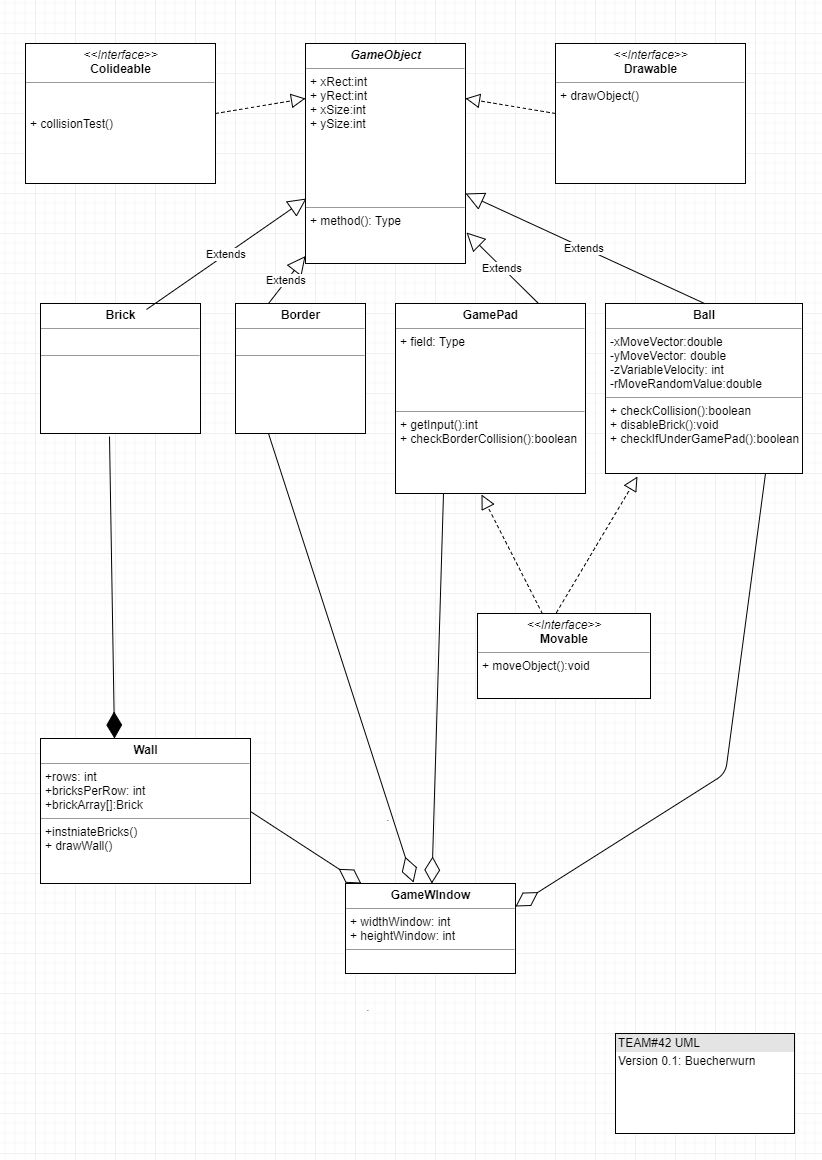
Also, we did not immediately manage to agree on the choice of controlling the paddle between the keyboard and the mouse, but eventually chose control from the keyboard.

Due to the fact that not all the members of the team were very familiar with the Git and the patterns, after a few days the work stopped - everyone was immersed in the study.

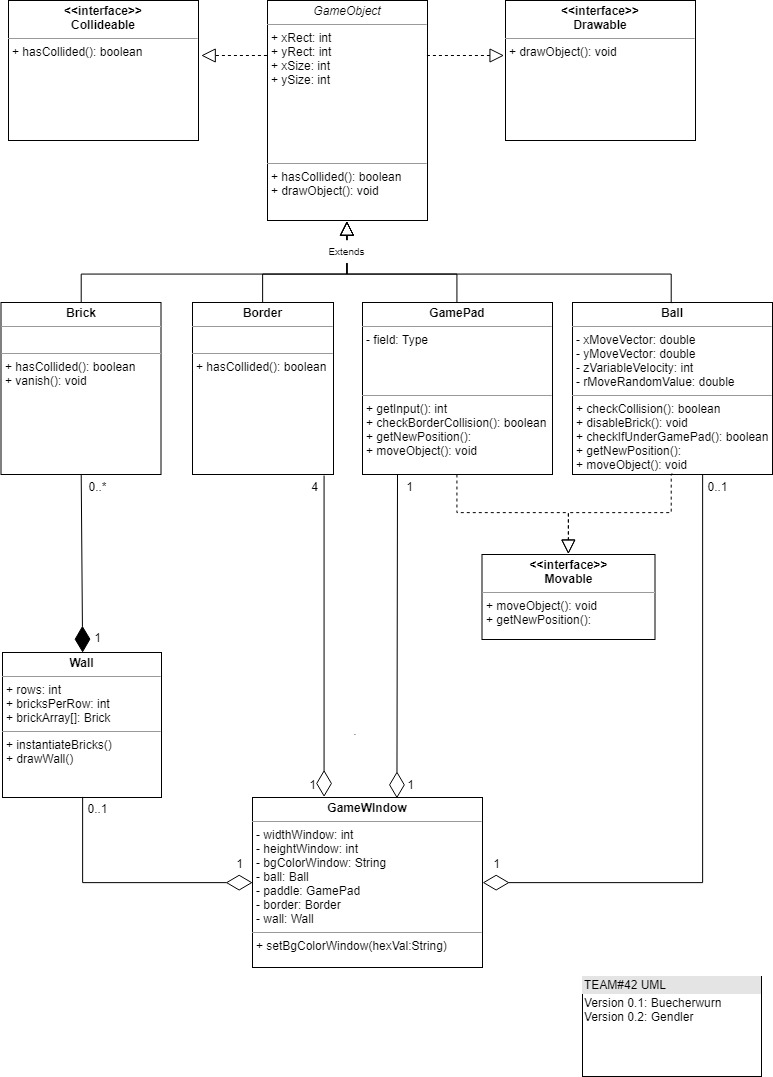
By the time the work was resumed and we had returned to the creation of the UML diagram, almost half the project time had passed.

Working on the diagram took some of the remaining time.

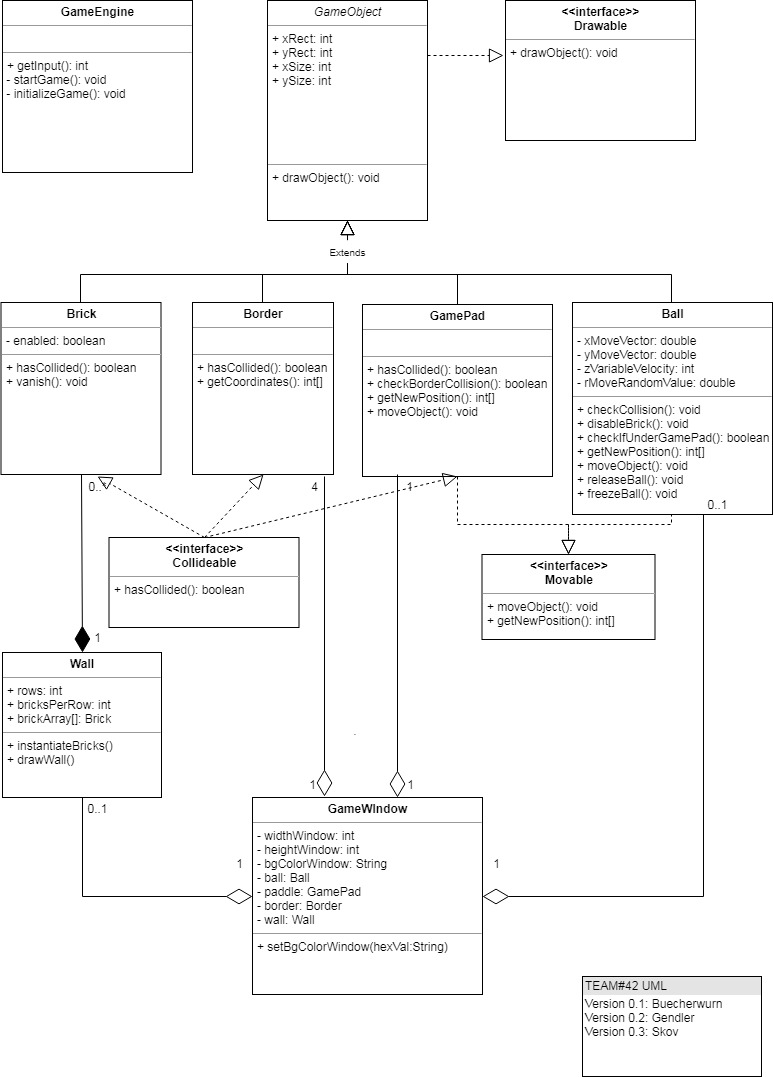
5 variants of the diagram were prepared. Its evolution is reflected in pictures 6 - 10.



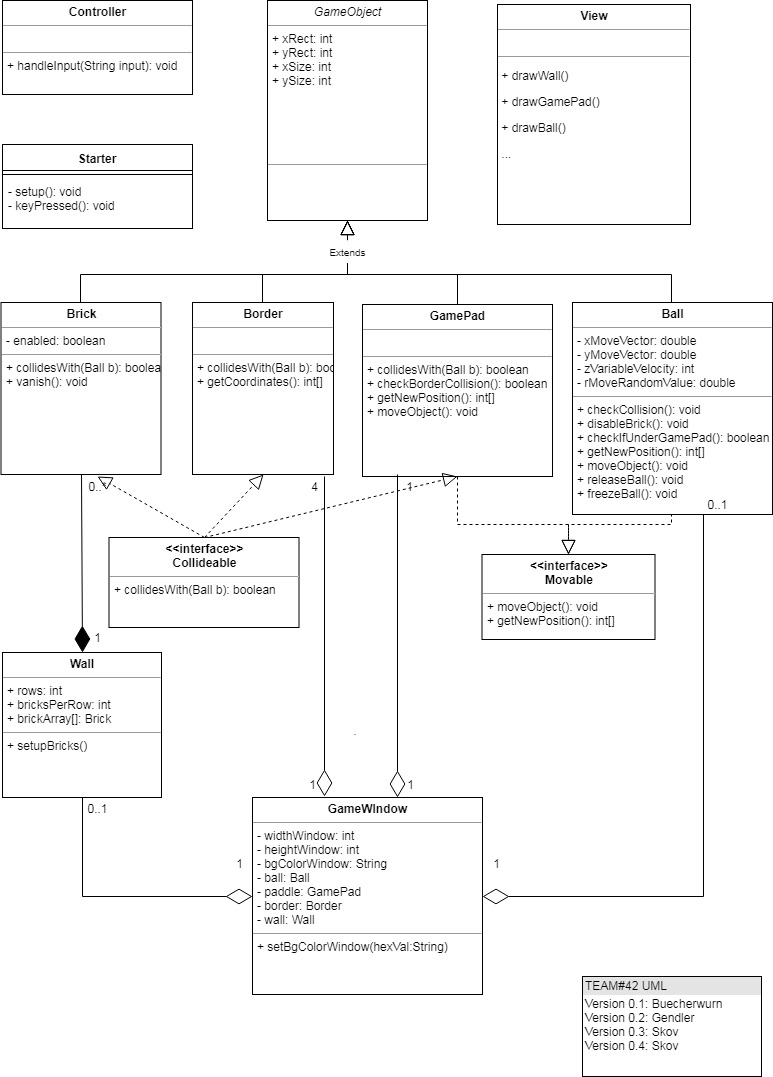
Picture 6. UML diagram - v1.



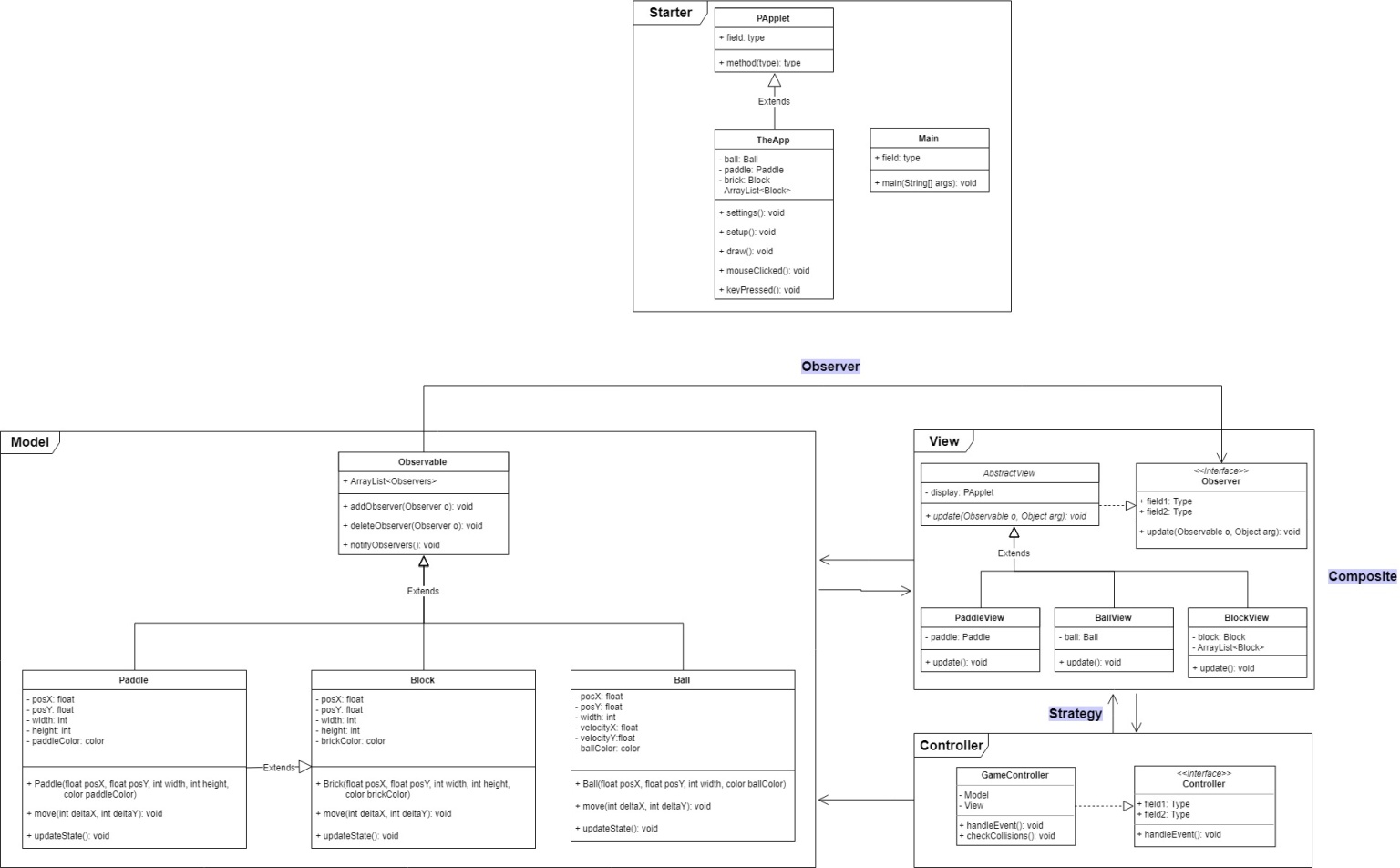
Picture 7. UML diagram – v2.



Picture 8. UML diagram – v3.



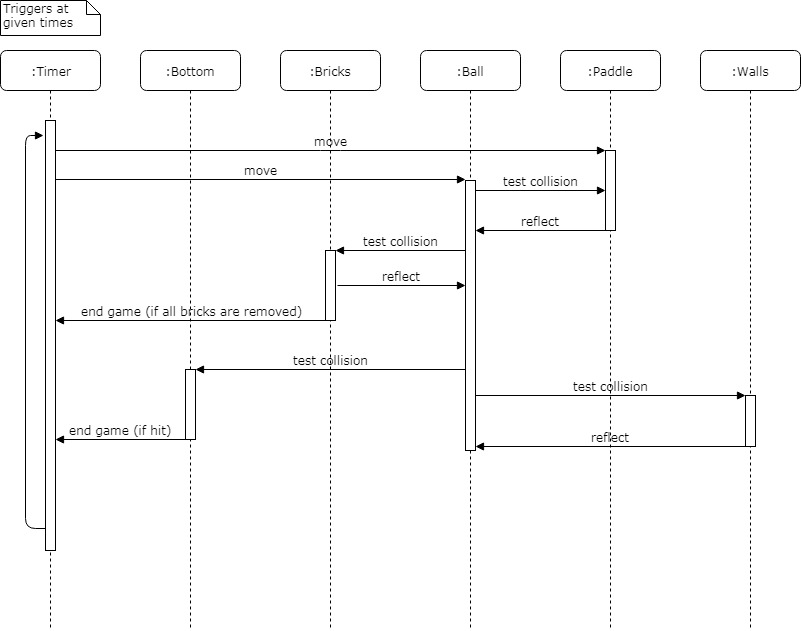
Picture 9. UML diagram – v4.



Picture 10. UML diagram – v5.

This version of the diagram was also published on the general forum.

In addition, at one stage of the work on the class diagram, a Sequence diagram was also developed (pict. 11).



Picture 11. UML sequence diagram.

Finally, we came to the conclusion that time had almost come to an end, and almost all of it was spent studying new material, preparing the diagram and planning the game. And we had a very little time left for coding.

That is why, it was decided to take the existing code[[2]](#footnote-2) as a basis and refactor it on our own according to MVC and other patterns introduced in the course material.

**Based on the screenshots, you might think that our project is identical to the borrowed by us original one, but it's not! We are not plagiating! Though the processing source is not developed by us, we decided to focus on the MVC-part and did our best to refactor it accordingly.**

Two of us worked on refactoring the code, and the third was preparing documentation for the project.

The result of the joint work we are sharing with you today.

**Thank you for attention!**

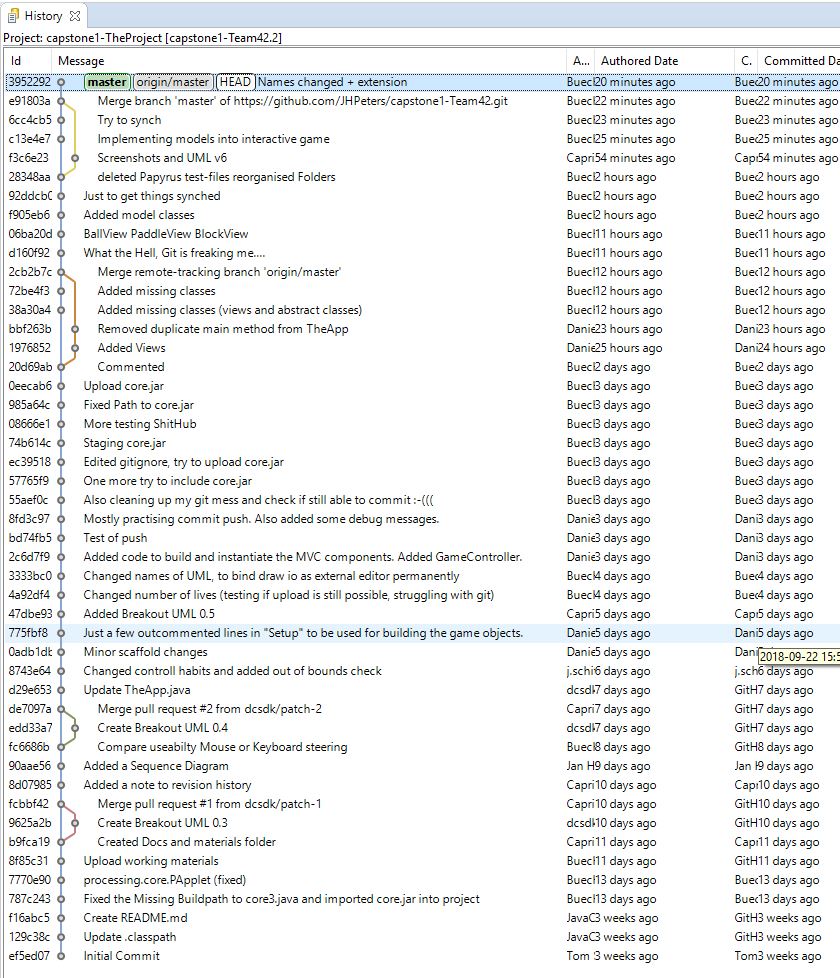
# Appendix:

**Illustrations:**

**I**) Current Version History Master Branch GIT

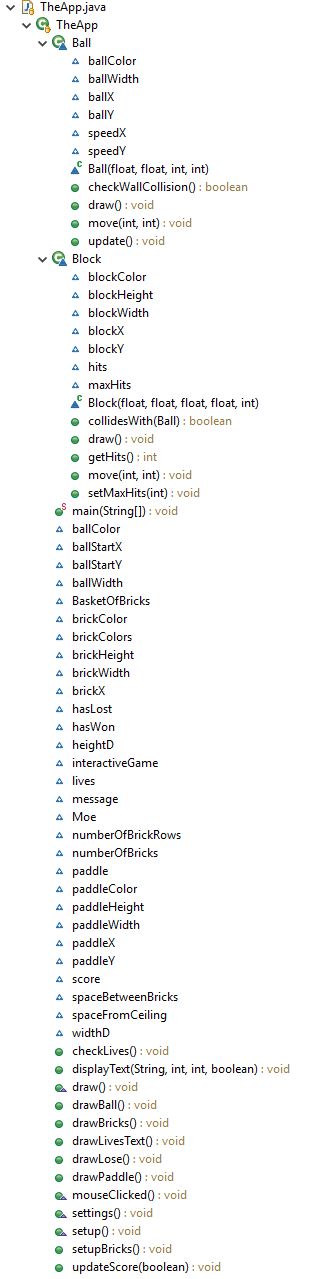
II) Currents structure of classic part (waiting for conversion to MVC)

III+IV) Current MVC-Structure

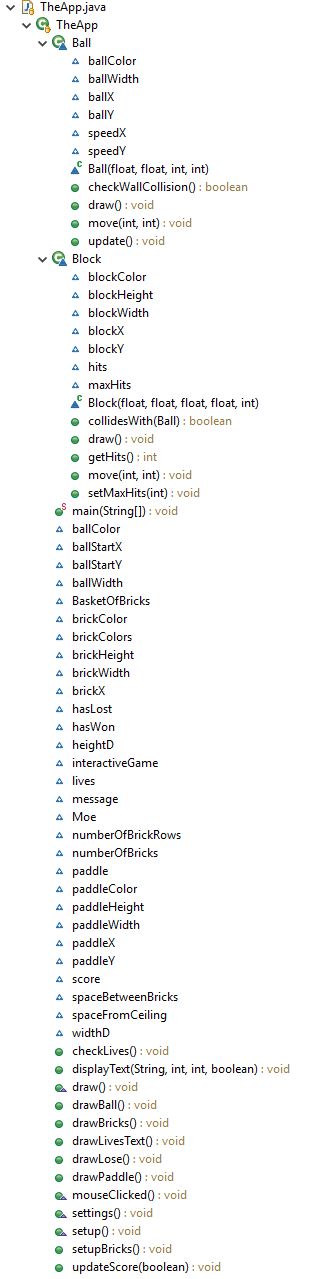


**I)**

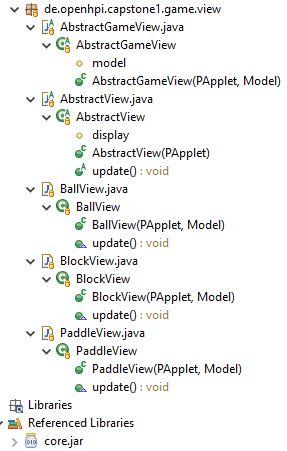
## Current Version History Master Branch GIT

**II)**

## Currents structure of classic part (waiting for conversion to MVC)

**III)**

## Current MVC-Structure

**IV)**

## Current MVC-Structure

1. https://en.wikipedia.org/wiki/Breakout\_(video\_game) [↑](#footnote-ref-1)
2. <https://www.openprocessing.org/sketch/104480> [↑](#footnote-ref-2)