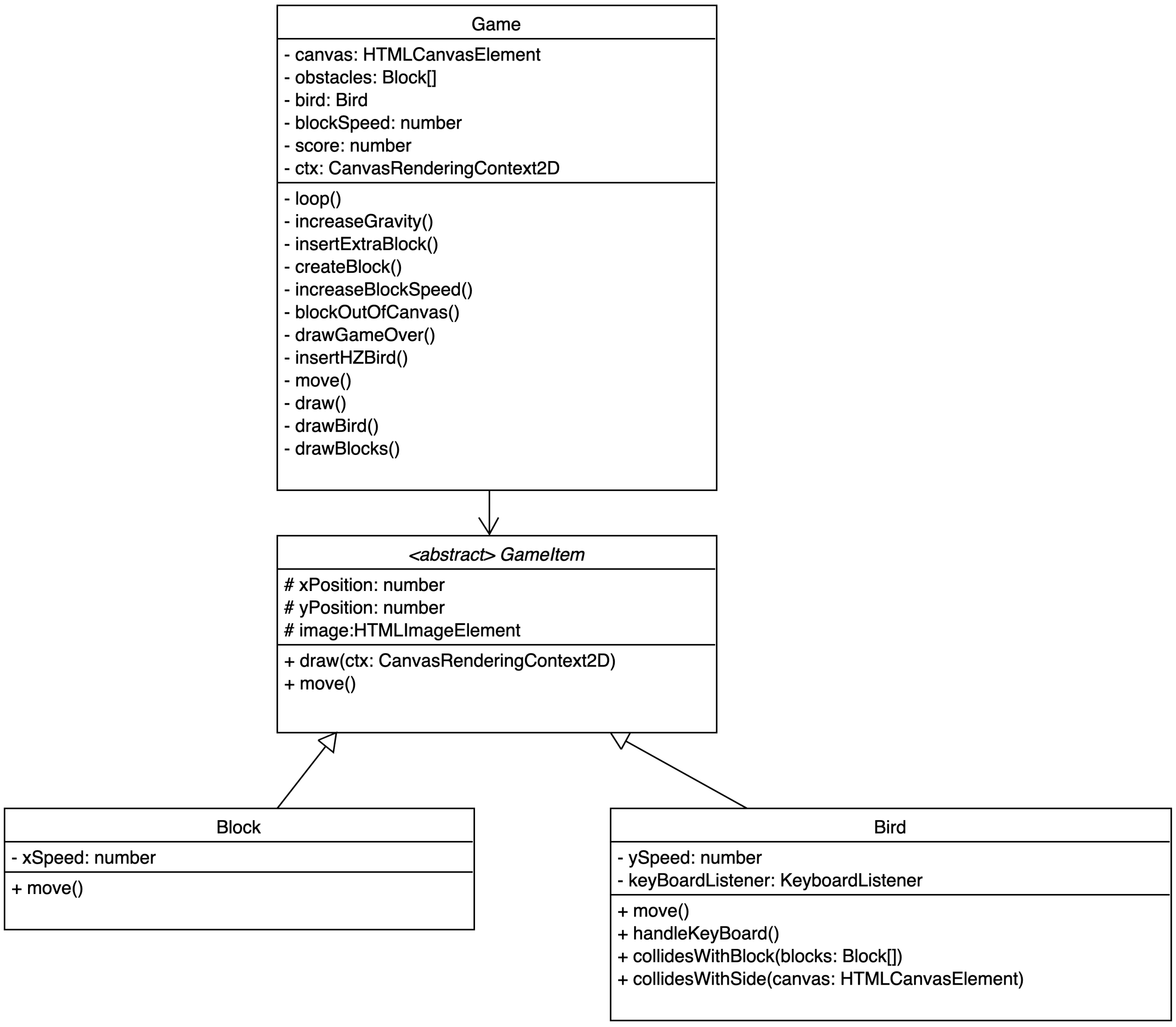
Case study Object oriented programming

- Start: 09:00 uur
- End: 12:00 uur
- Some students are entitled to have some extra time. They all have 20% extra time, for this exam it is 36 minutes in total. You can hand the answers in until 12.36.
- Tools you are allowed to use: internet, homework assignments and examples from the course. You cannot use social media and you are not allowed to ask any anwers or code from your classmates.
- Ownership: the code you hand-in is written by you.
- You can hand-in your assignment via HZ Learn. Hand-in your project as a **zip-file**.

Case

A friend of yours has recently been hired at a software-development company. This company wants to improve its market value by making serious games for students. However, nobody in the company has any experience with the development of games. Therfore they gave your friend the assignment to investigate how game development works and to make a first proof of concept of a game. And it worked! Your fried made a game that the company wants to sell. However there are some features that need to be added, but your friend did not account for this. Every attempt to add another feature to the game has failed.

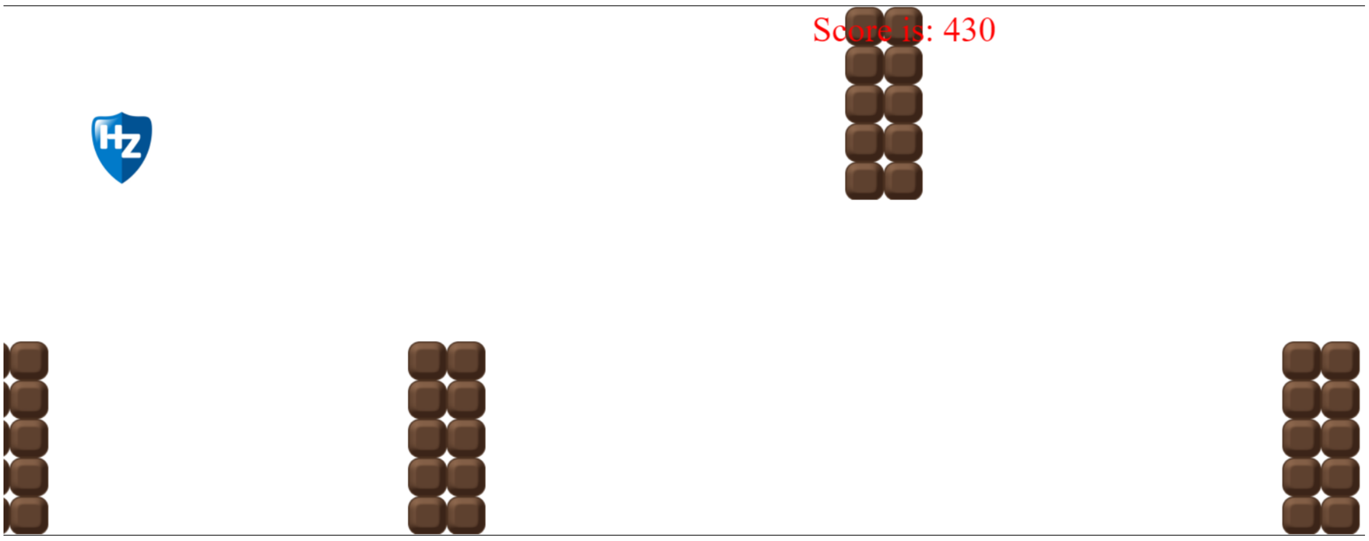
Luckily you followed the Object Oriented Programming course and you suspect what's wrong. You open the code and your fear becomes reality: almost all the code is packed into one class which means that everything depends on everything. After about half an hour of thinking and drawing you come up with the following design.



Now your friend is calling you in panic: the game has to be finished tomorrow! He promises you half of his salary if you can help. Can you implement the redesign so new features can be added?

The Game

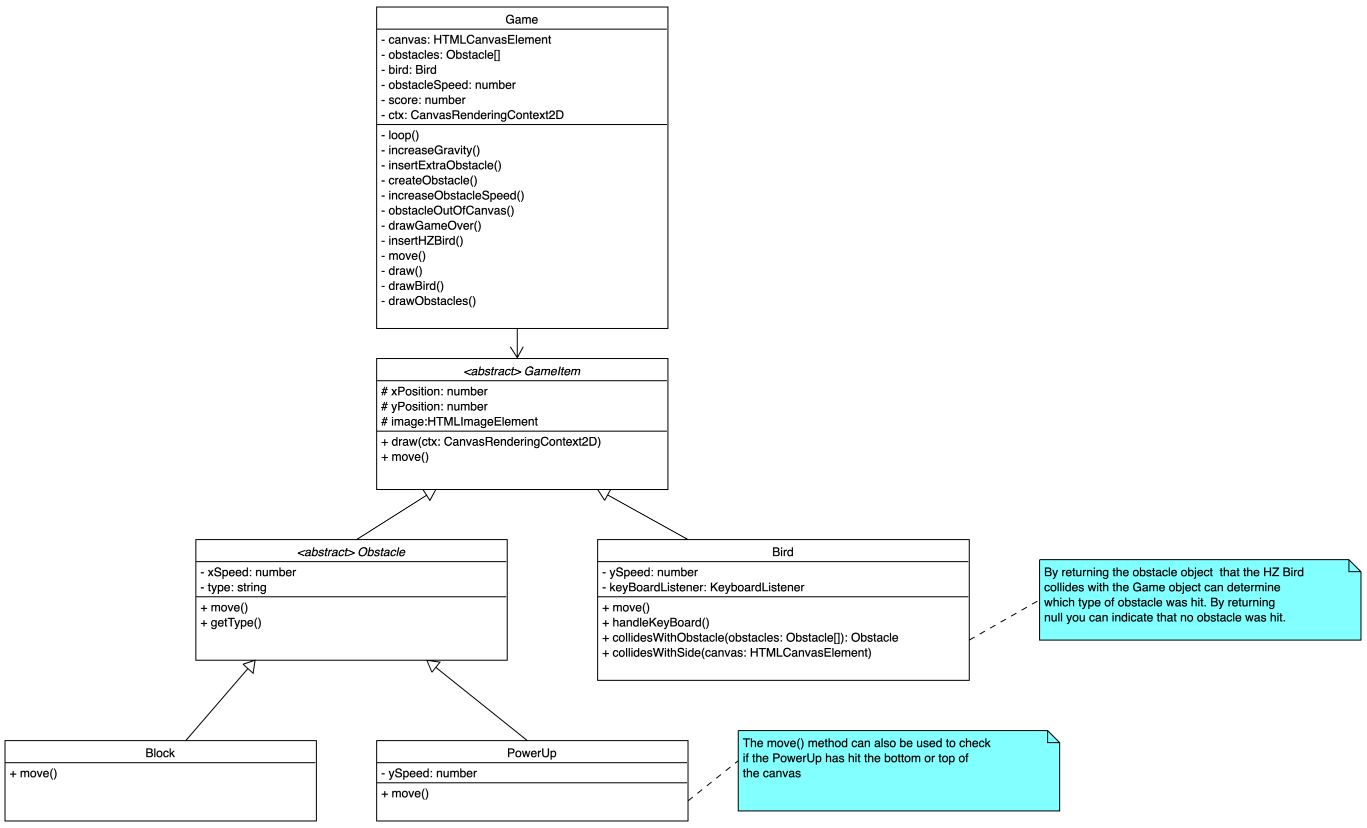
"Flappy HZ" is a game in which the player has to float the HZ logo between the blocks comming towards it. You can control the logo using the up-key to stop the logo from falling down. When the logo hits the side of the field or a block, the game is over.

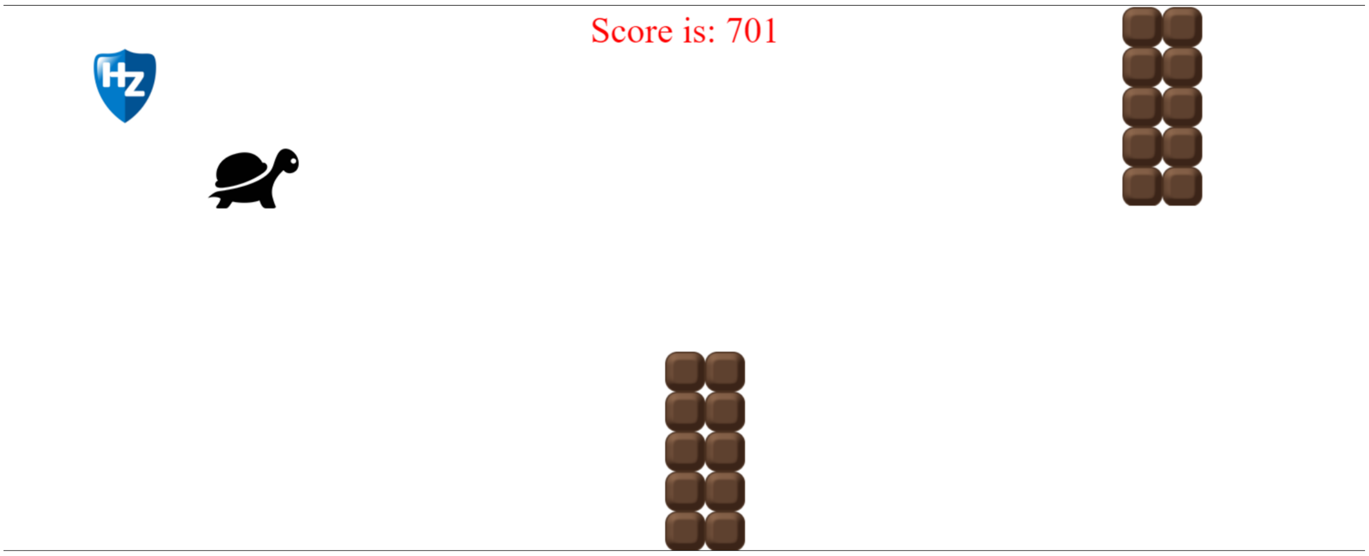


Assignment

Rewrite the provided code using the class diagram. Doing this succesfully leads to 1 point for criteria 4 and 6, and 0 points for criterion 8. In other words, your maximum grade will be a 7,0. You can score the remaining points by adding the following feature:

- Add a power-up to the game. Whenever the HZ logo touches the power-up (a turtle), the game will slow back down allowing the player to achieve a higher score. You can use the UML diagram below to implement this. The power-up will show up in the place of a regular block at random intervals. Next to moving to the left like a regular block the power-up will bounce up and down between the top and bottom of the canvas (see the HZ bird code for collision detection example).


Game
- canvas: HTMLCanvasElement
- obstacles: Obstacle[]
- bird: Bird
- obstacleSpeed: number
- score: number
- ctx: CanvasRenderingContext2D
- loop()
- increaseGravity()
- insertExtraObstacle()
- createObstacle()
- increaseObstacleSpeed()
- obstacleOutOfCanvas()
- drawGameOver()
- insertHZBird()
- move()
- draw()
- drawBird()
- drawObstacles()
<abstract> GameItem
# xPosition: number
# yPosition: number
# image:HTMLImageElement
+ draw(ctx: CanvasRenderingContext2D)
+ move()
<abstract> Obstacle
- xSpeed: number
- type: string
+ move()
+ getType()
Bird
- ySpeed: number
- keyBoardListener: KeyboardListener
+ move()
+ handleKeyBoard()
+ collidesWithObstacle(obstacles: Obstacle[]): Obstacle
+ collidesWithSide(canvas: HTMLCanvasElement)
By returning the obstacle object that the HZ Bird collides with the Game object can determine which type of obstacle was hit. By returning null you can indicate that no obstacle was hit.
Block
+ move()
PowerUp
- ySpeed: number
+ move()
The move() method can also be used to check if the PowerUp has hit the bottom or top of the canvas



ATTENTION: it could happen that during the implementation you make other choices than shown in the class diagram. Your grade will not be lower if your implementation deviates from the class diagram, however your implementation must still meet the criteria listed below.

Tips

- Take your time. Make small adjustments and test them before you proceed with your next change. This way you are always only one revert (ctrl + z) away from a working version of your code.
- Start by implementing one class.
- Make sure that after implementing your class your code has no errors and works correctly.
- Implement the next class and repeat this until you have all classes.
- The methods in the class diagram have no return type. You will have to add them in your code!

Hand-in

You always hand-in a zip-file (not a rar or any other kind), which contains **only** your src folder.

Criteria

Nr	Criteria	Points
1	student applies indentation consistently for readable and structured code	0,2
2	student uses a standardize commenting style according to the AirBnB style guide	0,4
3	student naming styles for attributes, method and classes (alongside with variables and functions) is consistent and following the AirBnB style guide.	0,4
4	student applies classes (attributes and methods) to structure the code	2
5	student applies inheritance to avoid code duplication	2
6	student adds types to attributes, parameters, return values explicitly.	2
7	student applies encapsulation in a class to hide the data and creates an interface for other objects	1
8	student applies polymorphism to avoid redundancy	1

Nr	Criteria	Points
9	student applies DRY principles to reduce complexity and stimulate readable and maintainable code parts	1