

UML Codex Naturalis GC10

Riccardo Polelli, Filippo Pozzi, Federico Quartieri, Giacomo Tessera

2024

Indice

1	Model	2
1.1	Card	2
1.1.1	Challenge	2
1.2	GameState	4
1.3	Enumerations	5
2	Controller	5
3	View	5
4	Complete UML	6

1 Model

1.1 Card

In this section we will shortly explain how we structured the classes regarding the various types of cards.

Here is a table that shows what all the types of cards have in common, so that it is easier to aggregate them into classes and subclasses.

Card	Has Corner (4)	Has Resource Corner (max 4)	Has Item Corner (max 1)	Has Points	Has Challenge	Has Resource Needed (max 5)	Has Back Resource
Resource	x	x	x	x			x
Gold	x		x	x	x	x	x
Starter	x	x					x (max 3)
Objective				x	x		

The Cards are never discarded, they are always used for something.

1.1.1 Challenge

1.1.1.1 Structure Challenge The structure challenge is only for objective cards.



1.1.1.2 Resource Challenge The resource challenge is only for objective cards. The type of resources needed in the objective cards can vary.

In the objective cards is like this:



1.1.1.3 Item Challenge The item challenge is only for gold cards.

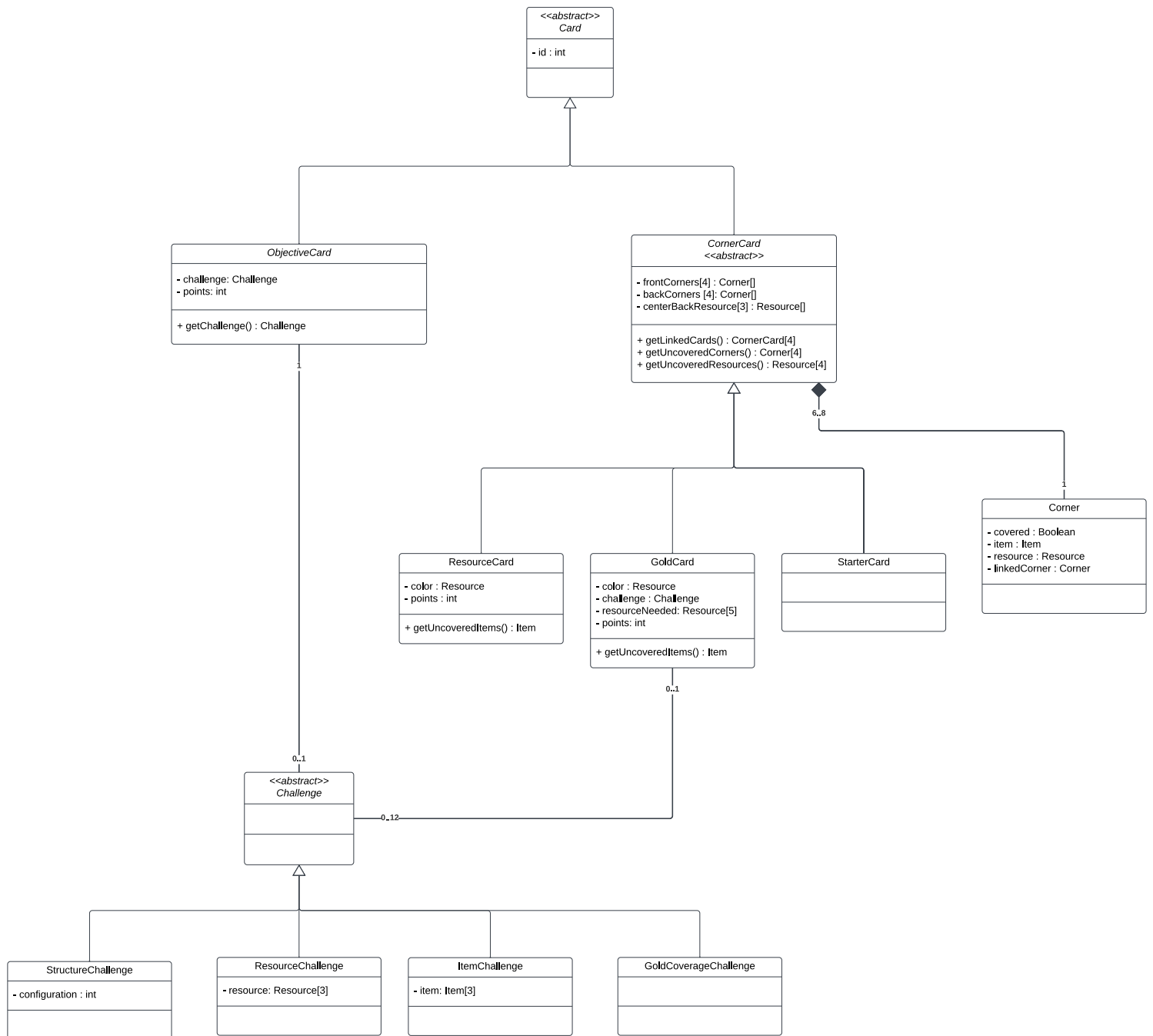
In the gold cards is like this (the one on the top of the card):



1.1.1.4 Coverage Challenge This challenge is only for gold cards. Is the one on the top of the card.



Here is the UML for both the Card and the Challenge:



StructureChallenge and **ElementChallenge** are used in **Objective**.
GoldCoverageChallenge and **ElementChallenge** are used in **GoldCard**.

Note that challenge can be null. **challenge** in the **GoldCard** is the challenge to do in order to do point. Also note that for the **GoldCoverageChallenge** the value of **points** is always 2.

The **configuration** can have any type since the number of configuration is limited, **int** is probably the most practical.

About element:

- In **goldCards**, `Element.size() == 1`
- In **objectiveCards**:
 - example of 3 animal: `element=[animal, animal, animal]`
 - example of 1 Quill and 2 Inkwell: `element = [Quill, Inkwell, Inkwell]`

Probably we should give an id to all of the cards so that we can render them with the right texture and we can track which one has been used or not.

`covered` is true only if is under another corner.

Order from top-left to bottom-left [0-3]

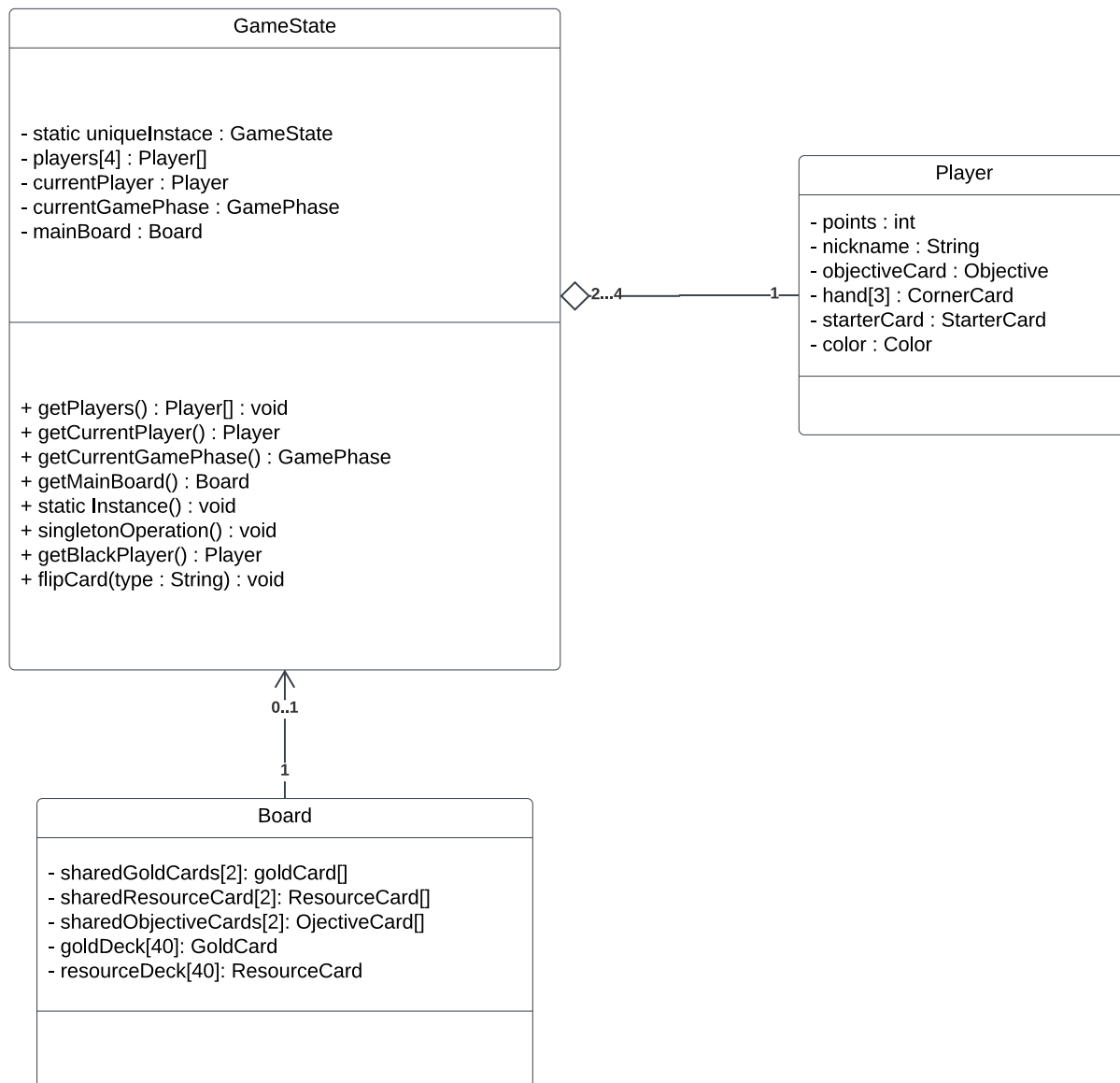
01

32

The arrays have a null value if the corner does not exists.

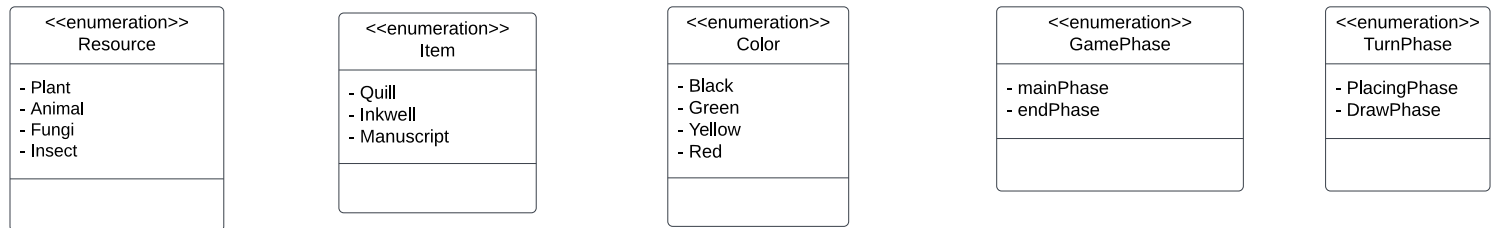
1.2 GameState

The first player in the data structure is the black one, something like `public Player blackPlayer() { return player[0] }`. Note that a **round** is made up by 4 **turns**.



1.3 Enumerations

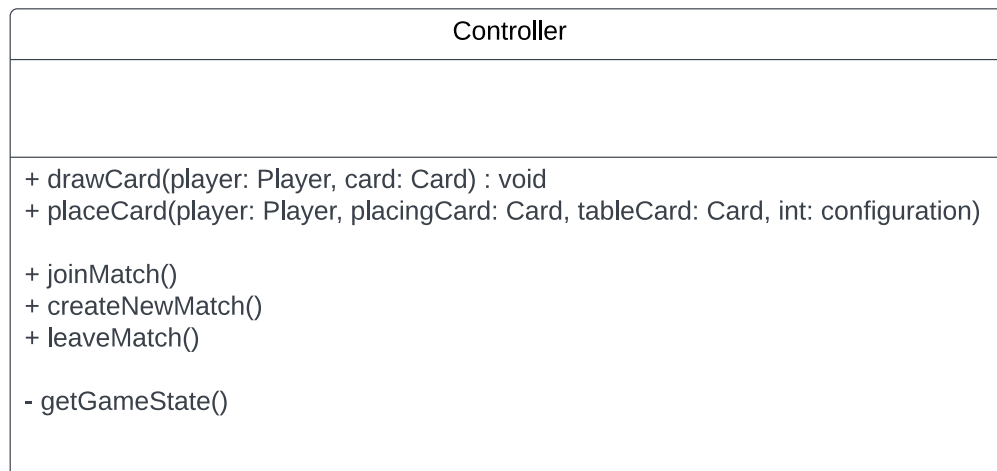
We will use some enumerations to define our data types. We cannot show all the association with the other classes because graphically it would too confusing. Just remind that these object are actually associated with the classes that make use of them.



2 Controller

We started implementing the Controller while we are still studying how to implement the View, so it is still *wip*.

In `placeCard` you can retrieve the information of a multiple corner placed card by checking in the method, you just need to place it on one single card, even if it will affect another card.



3 View

wip

4 Complete UML

