# HR the gathering

A new amazing trading card game from Hogeschool Rotterdam

The following is the scenario for a trading card game
Only some parts of the game are described and need modelling

INFSAD team
4 March 2024

## The goal of this assignment:

The goal is to apply UML and design patterns to model the engine (NO UI) of the trading card game HR the gathering and translate the diagrams into code. Your code must respect your design (and viceversa).

## Description of the assignment

You are going to *design* and *implement portions* of this game.

The main structures that will hold the logic (but not all the logic) will be your main objective. The structure must be built with future in mind, how can it be expanded from a third party later without putting too much hands on your own code. You should make full usage of most of the topic described and used in this course and apply them to your structure.

## General scenario

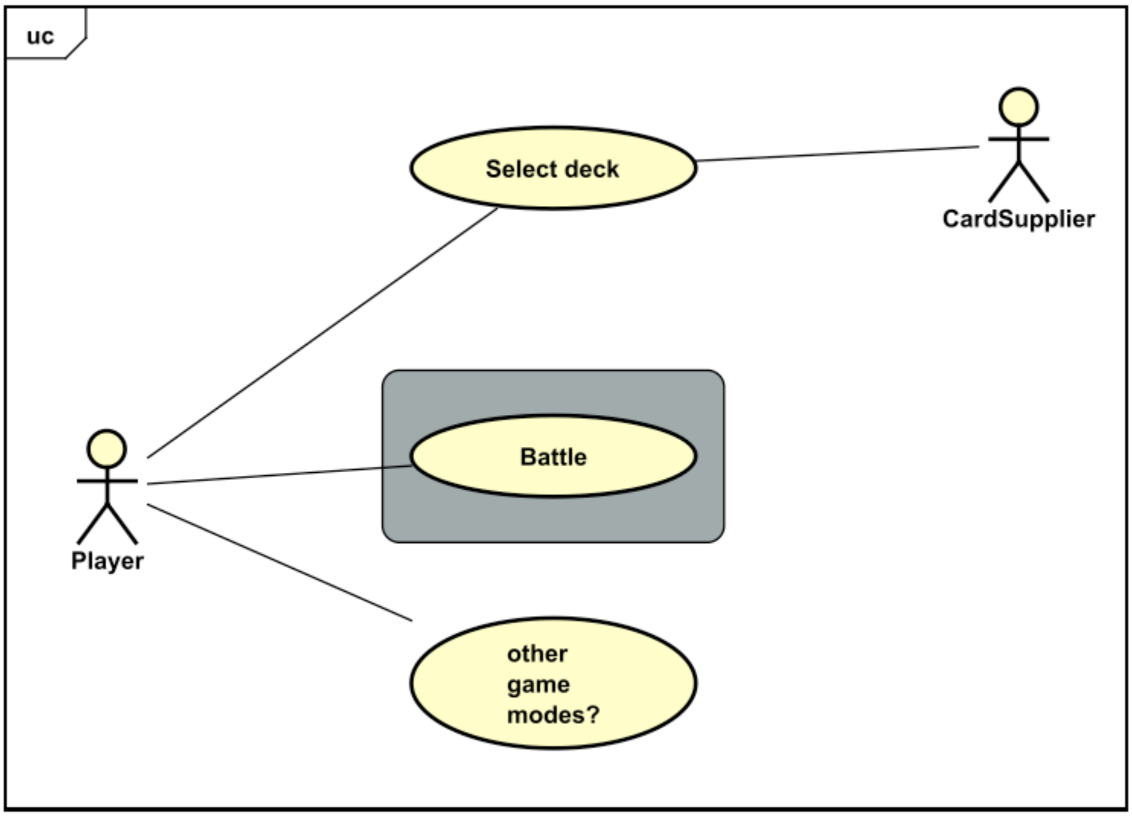

Fig.1 Generic use case diagram. You will be implementing the “battle” that is highlighted in grey.

Imagine a trading card game, similar to Magic the gathering™, Pokemon™ trading cards, Yugioh™, or Hearthstone™. If you never heard or tried have a look at some of the videos present online that shows the gameplay.

We want to create a game that has some of the features of these "traditional" trading card games.

To be specific, a game where we have different kind of cards, some of them have effects and others do not, but all are collected (by each player) in their won decks. The decks consist of cards collected from a big pool of cards, they can be traded and the pool can be extended with expansions that can alter the gameplay.

## Game Rules

The game is played by 2 players (or wizards) at times. Both players own their deck from which they can draw cards. The game is played in turns but some card/effect can be played in the opponent turn. Each card is a spell.

Each wizard extracts spells from his deck.

Some spell generate permanents other do not (more details later).

All the spells require a cost that is payed by using lands.

Each player can have their own hand made of their own cards, up to 7 cards, or at the end of the turn the player must discard the excess in the discard pile. Each discard pile collects all the consumed or discarded cards.

Each players draw cards and play them from their hands to apply effects on the opponents and to the battlefield (where the permanent cards are placed).

Each player has his own life that can be altered by spells/effects or activities in the game. Each player starts with a value 10 for his life.

Each player take turns until one of the 2 player wins.

## Winning condition:

The other opponent must be left with *no life* (at any time) or with *no card* left to draw from the deck (the loss will be determined at the moment of drawing a card of the deck, be it the result of a card effect or because of the standard turn start), or the opponent is giving up on the game.

## Cards:

All cards can be made of parts:

- Activation effect
- Cost to be played,
- the kind of the card:
  - Cards can be of different kind
    - land
    - spells
      - Spells can be instantaneous or generating permanent(s)
      - Permanents can be creatures that can attack
- Effect(s)
- Cards may have different colours
  - Red, blue, brown, white, green

Lands are permanents that do not have a card cost.

Cards can have up to 3 copies in each deck, no more.

In the future we may need some permanent may be color-less.

Some spells become permanents (also called permanent cards, ex: creatures) or are instantaneous and be discarded just after.

Permanent cards, once played stay in the game until destroyed or consumed. The instantaneous are always discarded after usage.

During each turn the players can interact with each other by means of permanent effects or instantaneous cards. Only instantaneous cards can be cast outside the owner turn.

Cards (beside lands) contains an activation effect or cost, and some permanent effect.

## Effects:

An effect is an action that is started when the card is successfully played.

The effect/s can stay in place according to the description, it could last one turn, or longer. At some point the effects are removed (if it was for one turn for example: at the beginning of the next il will reset, if it is a lasting effect, will stay until the card with the effect is destroyed).

Let's have a look at a spell card has an effect that when casted gives a creature an alteration (permanent or temporary) of the stats (attack/defence).

Temporary effect example: an instantaneous spell costs 3 energy, is a green spell that gives +3/+3 at the creature till the preparation phase. At the preparation phase of the owner, the creature will cease to have the effect of this spell.

Permanent effect example: an instantaneous spell, it costs 4 energy and has an effects as the following—>"while this creature is on the board, all creatures that belong to the owner of this one will have +2 in defence". When this creature is destroyed or killed the effect will no longer reach the creatures and disappear.

The effect is played automatically when the card is played, independently from the kind of card. Effects can influence players, other cards in play or the hand of a player. Effects can influence phases of the game and can modify so the gameplay.

The effects that require a target can't be activated if the target is not legitimate (ex.: if a card affects a land but no lands are present the effect will not take place).

Instantaneous cards are considered cards that play automatically the effect/s and, after that, are discarded.

Instantaneous cards may have effect on each other (ex.: counter an instantaneous spell) and they can chain together. The resolution of the cards will be according to a stack system where the first in is the last to be resolved (see this article for further details: https://www.boardgamebarrister.com/magic-what-is-the-stack/ ).

### Playing a land:

A land can be played as a spell, but with zero cost, no interruptions or effects can take place while the land is placed.

Obtain lands energy:

A land can be turned 90° to obtain one unit of the appropriate kind of energy (each land can generate 1 energy of the colour of the land). The energy point is accumulated in the energy reserve of the player, till the end of the turn. If a land is turned, it can't be reused till the next beginning of the owner's turn, when lands are put back in their original position.

Energy is obtained only when you need to cast a spell or use it for some other purpose, there is no "using a land" or accumulating energy that will not be spent immediately.

Lands can be used only after passing the first [*preparation*] phase.

### Casting a spell (playing a card):

The amount of cost in lands energy must be available at the time of the casting.

### Creatures:

A creature is still a spell that generates a permanent creature on the game floor. A creature is a permanent of a specific kind and colour. It can have effects. It must have attack value and defence value (ex: 3/4, 3 attack 4 defence). Attack can be 0 but never less, if at any time the value of the defence goes at 0 or under, the creature is placed in the discarded pile.

## The attack:

After an attacker is being declared it can't defend from other attacks until the reset in the next owner turn.

If no defender is declared (from the opponent), the creature attacking removes from the life of the opponent a value equal the the current attack value.

If a defender is declared, the damage is inflicted to the defendant creature subtracting from the defence value (as by definition of creature if the defence value goes to 0 or less the creature is discarded). Whatever the result, the defending creature removes from the attacking creature defence value a number equal to the defending creature attack value.

# Structure of the gameplay

## Preparation:

Each player shuffle its own deck and 7 cards are dealt from the respective decks to their owners.

## Each turn is divided in phases:

1) [*preparation*]: At each beginning, temporary effects are erased and reset to their original states, for example: turned lands come back to normal. In this phase other effects of cards can take place (if they have effects that works in this phase).
2) [*drawing*] The player gets a card from his deck, and add it to its hand if the player can't get a card, will loose.
3) [*main phase*] The player may play card(s) according to the availability of resources in play (only cards that have sufficient energy obtained from the lands can be played).
   - [*attacking*] (this is done during the *main phase*) The player **may** decide to attack the opponent with the creatures cards in place that are

able to attack, once attacked the creature cards are marked as used (as the lands). Only creatures cards can “attack”.

4) [*ending*] The turn ends and the player must discard the cards in his hand till at most 7 are present. If the player has less than 7 cards, nothing happens.

## An example of a small play (a few turns):

2 users are playing.

Each user has a deck of 30 cards and a discard pile of 0 cards.

Each user has no permanents in the game.

Each user has 7 cards in hand and 10 lives.

Turn 1.

User A (Arold) : starts to play his turn first.

1) gets a card from the deck.
2) He has 8 cards
3) Plays 2 lands (one after the other)
4) He can’t play any card as the land can’t be used the first turn that are in place.
5) He has now 6 cards, and ends the turn.

*End situation: 6 cards in hand, 2 lands on the floor, full life.*

*End situation: 7 cards, no permanent on the floor, full life.*

User B (Bryce):

1) gets a card from the deck.
2) He has 8 cards
3) Plays 1 land
   1) He can’t play any card as the land can’t be used the first turn that are in place.
4) He has now 7 cards, and ends the turn.

*End situation: 7 cards, 1 land on the floor, full life.*

*End situation: 6 cards in hand, 2 lands on the floor, full life.*

Turn 2.

User A (Arold):

1) gets a card from the deck.
2) He has 7 cards
3) Plays 1 land
4) He takes 2 energy from his lands, that are now marked as used
5) He uses 2 energy from his reserve to cast a permanent (a blue creature, 2/2).
6) He decides not to attack with the creature just summoned.
7) The creature has the effect that removes a random card from the opponent hand when it comes into play.
    1) User B discards a card randomly.
    2) The card is put in his discards pile.
8) He terminates the turn with 5 cards.

End situation: 5 cards in hand, 2 used lands on the floor, 1 land played, a permanent creature played, full life.

*End situation*: 6 cards, 1 land on the floor, full life.

User B (Bryce):

1) gets a card from the deck and does not play any card.
2) He ends the turn with no action done (nor attacks).

*End situation: 7 cards, 1 land on the floor, full life.*

Turn 3.

User A (Arold):

1) gets a card from the deck.
2) He has 6 cards
3) He decides to declare an attack with the 2/2 blue creature

1) Before the attack is concluded he gets 2 energy for the lands (1 used, 2 to be used) and casts a green spell that gives +3/+3 at the creature
2) User B interrupts the resolution by getting 1 energy from his lands and casts an instantaneous red counter spell that cancel the green spell just casted (one land used on the floor, the permanent creature is in considered attacking)
3) User A interrupts the resolution by getting 1 energy and cast a blu instantaneous counter that cancel the red spell
4) The resolution of the interruption stack is then applied, and the last blue instantaneous is resolved first, the second item in the stack is so removed, and the last green spell is then applied to the creature that becomes 5/5 and delivers 5 damages.
5) User B life goes to 5

End situation: 4 cards in hand, 3 used lands on the floor, a permanent creature played in a state of attack, full life.

*End situation: 6 cards, 1 land on the floor, 5 life.*

## Implementation:

To complete your assignment you must finalise the following implementation and the required diagrams.

Groups should be at most of 2 students.

As the starting point, we need an UML model that shows the essential structure and behaviour of the game. It should be the base line to give to a developer partner that could implement the game.

The behaviour shows how the game plays out. This can be done as text only sequence of printout of the turn run. We want to see the moves a player makes, its effects and the resolution of the effects in one big log listing (no need for files, just printout).

We ask you to implement the data structure in the most appropriate way and make an example of a situation described in turn 2.

**THERE IS NO NEED FOR A GUI. KEEP IT SIMPLE!**

**Implementation must only show the basic step by step and allow us to understand that you applied correctly the design/design patter for each element. YOU MUST NOT IMPLEMENT AN INTERACTIVE CONSOLE GAME.**

Suggestion: The classes and usages are all in the text, sometimes designing extra classes can help, in other occasions it might make your implementation very difficult. Always keep in mind the future (and the design suggestions we discussed in class)!

Suggestion: as the game is in an advanced stage (not in turn one), you can implement a method that “create current state” where the situation is initialised in the most appropriate way up to the beginning of turn 2 (cards already played, life values, cards in hands, etc…).

Suggestion: use interfaces/abstract classes to invoke the right dynamic type from the main game run.

You should use the most convenient design pattern for:

1) Creating cards and new expansions of an unspecified new type of permanent.
2) Creating and maintaining only one existing general board for the game, that collects all the status of the elements that are placed in it.
3) Imagine to have to keep updated the status of each card that comes into play, if your opponent has an effect that effect the card, all the affected cards should be updated accordingly. Create a generalised version that can sustain extensions with dedicated channels of communications for any kind of event.

**Suggestion: overall you should have 4 of the following design patterns implemented in the right place: Singleton, Factory, Pub-sub, Observer, Composite, State design**.

## EXTRA: SUGGESTIONS FOR THE DESIGN

THE OBJECTIVE IS THE DESIGN ASPECTS of the game, NOT the precise implementation of each card.

Focus on the abstractions of details in modelling the behaviour not on the details on its own. The perspective of the future extension should allow you to integrate code without rewriting your own classes that are already implemented.

**TLTR; If by adding an extensions you are going to have to reimplement classes, you are doing it wrong.**

**Try to highlight with colours the design patterns in the diagram (this will make our correction faster).**

## Files to deliver:

### UML diagrams

- A picture with a class diagram. It should contain all the elements that are implemented in the C# files: the class diagram should represent the implementation of all the entities involved in the required run.
- Each picture should have inside name and student number of each author
  - The name of the file should be represented as follows:
    - Ex: CD_stdnum1_student2_INFSAD.jpg (stdnum1 and stdnum2 are the student numbers, in case of lone delivery stdnum2 is not needed, png/jpg are ok as long as the level of compression allow to read the details).
- A picture of a Sequence diagram **ONLY of turn 2**
  - The name of the file should be represented as follows:
    - Ex: SD_stdnum1_stdnum2_INFSAD.jpg (stdnum1 and stdnum2 are the student numbers, in case of lone delivery stdnum2 is not needed, png/jpg are ok as long as the level of compression allow to read the details).
- A picture of a Finite State machine: state of a card
  - The name of the file should be represented as follows:

- Ex: SMD_stdnum1_stdnum2_INFSAD.jpg (stdnum1 and stdnum2 are the student numbers, in case of lone delivery stdnum2 is not needed, png/jpg are ok as long as the level of compression allow to read the details).

## Files C# to be submitted

The implementation of all the classes in the class diagram and the run described in the Sequence diagram.

Files names should reflect the structure of your program representation, good/common praxis should be respected.

Inside each file there should be all the data of each author in a comment.

Ex:

```
/*
Studentname 1  studentnumber1
Studentname 2 studentnumber2
*/
```

## Zip file

Only zip (no rar/7zip). The global zip file should be named as (stdnum1 and stdnum2 are student numbers): **INFSAD_stdnum1_stdnum2_2324.zip**

## Delivery:

Deadline: April 8, 2024, 11:30 PM

## Evaluation:

To be sufficient:

- Must contain all the elements requested, each diagram should have the requested depth to understand the structure and the general behaviour from the assignment description.
- Must be coherent with the implementation required (all the classes descriptors should be present in the diagrams as they are in code, we are not interested in the body of functions, but in the structure).
- Must be correct in all its parts, the code should be executable correctly and the diagram should not contain errors in the syntax.
- The code should be cross-platform.
- The design patterns and structure of the concepts explained in class should be correctly implemented in diagrams and in code (using design patterns as checklist or just to have them there with no real usage or motivation is considered a wrong application).
- Each file should be appropriately labelled **in** the picture/file with the data of the author/s
- All the pictures should have sufficient clarity and definition to understand the content.
- The solution produced should compile with no problems/warnings.
- No external libraries/installation lib should be needed

- **<u>The team of the assignment may be requested to present the solution. It is very crucial both students be confident all the code, modeling and design patterns.</u>**

Failing to meet any of the above requirement will make the evaluation insufficient.

Checklist FOR THE STUDENT:

CHECK IF YOUR NAME IS PRESENT ON EACH FILE DELIVERED!

*.cs files: C# files for all the classes (also possible multiple classes in the same file).

The solution does not produce errors, there are no dependency from external libraries.

The solution should contains all the design patterns that are needed and they are correctly implemented.

Picture of class diagram

check correctness of the syntax

check if design patterns are applied properly

check if all the entities have the right elements in it and correct types

check if the picture has all the data of who made it

Picture with State machine diagrams with all the details and transitions

check correctness of the syntax

check if design patterns are applied properly

check if all the states have the right elements and right transitions

check if the picture has all the data of who made it

Picture with the sequence diagrams for turn 2

check correctness of the syntax

check if design patterns are applied properly check if all the entities have the right elements and right calls

check if the picture has all the data of who made it

Check if all the file names of all the files delivered have the necessary format.