HR the gathering

A new amazing trading card game from Hogeschool Rotterdam

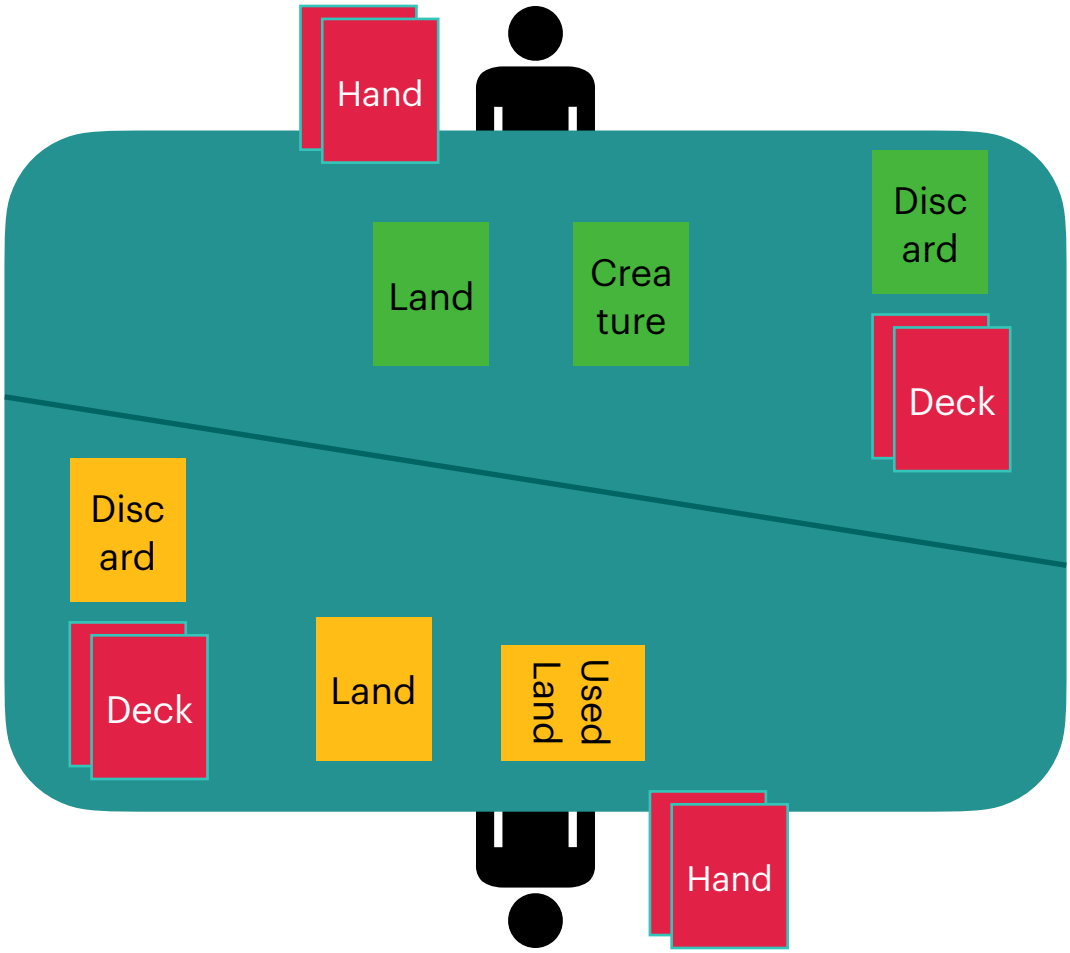
The following is the scenario for a trading card game
Only some parts of the game are described and need modelling

INFSAD team
3 February 2023

The goal of the exercise:

The goal is to apply UML and design patterns to model the structures (NO UI or no real game, meaning that classes can be close to empty) of the trading card game “HR the gathering”, and translate the diagrams into code. Your code must respect your design (and vice versa).

The focus is on the structure and the code design.



Description of the exercise

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 the **future** in mind: how can the structure be expanded from a third party later without putting too many hands on your code?

You should make use of most of the topics described in this course and apply them to your structure!

General scenario

Imagine a trading card game, **similar (not the same)** to Magic the gathering™, Pokemon™ trading cards, Yugioh™, or Hearthstone™. If you have never heard of or tried such games, look for some online videos showing the gameplay (https://en.wikipedia.org/wiki/Collectible_card_game, https://www.youtube.com/watch?v=aj9AlyvlJ5w, https://www.youtube.com/watch?v=Ym53BuJXe3Q).

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

In this game, we have different *kinds* of cards: some of them have *effects,* and others do not, and each player collects them into their decks. The decks consist of cards collected from a bigger card pool; cards 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 (called wizards) at times. Both players own their deck from which they can draw cards. The game is played in turns but some cards/effects can be played on the opponent's turn. Each card can be a spell or a land. Each wizard extracts the spells and cards from his deck to be played.

Some spells generate *permanents* others do not (more details later).

All the *spells* require a cost that is paid by using lands. Lands do not have a cost.

Each player draws cards and plays them from their hands to apply effects on the opponents and to the battlefield (where the permanent cards are placed).

Each player's hand consists of up to 7 cards. At the end of the turn, the player must discard the excess in the discard pile.

Each discards pile collects all the consumed or discarded cards.

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

Each player takes turns until one of the 2 players wins.

Winning condition:

The other opponent must be left with *0 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, being it the result of a card effect or because of the standard turn start).

Cards:

All cards can be made of parts:

- Activation effect
- Cost to be paid (lands have no cost, not zero)
- Kind of the card:
  - Cards can be of different kinds:
    - lands,
    - spells:
      - Spells can be instantaneous or generate permanent(s)
        - [***Artefacts*** (see expansion) are special kinds of permanent]

      (you can ignore this until you get to the expansion)
        - Some permanent can be creatures that can attack
- Effect(s)
- Cards may have different colours
  - Red, blue, brown, white, green *(and now neutral from the expansion)*.

Lands are permanents that do not have a card cost.

Some spells become permanent (also called permanent cards, ex: creatures) or are instantaneous and 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 using permanent effects or instantaneous cards. Only instantaneous cards can be cast outside the owner's turn.

Cards (except for lands) contain an activation effect or cost, and some permanent effect.

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

Reflection: In the future, we may need some permanent to be colourless. Will this affect your code? And if so, how?

Casting a spell (playing a card):

The player must have available the amount of energy sufficient for casting the spell

Playing a land:

Land can be played as a spell, but with no cost. No interruptions or effects can affect its placement.

Obtain lands energy:

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 card 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.

Creatures:

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

The attack:

After an attacker (creature) is declared, it can't defend itself from other attacks until the reset in the next owner's turn.

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

If a defender is declared, the damage is inflicted on 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's defence value a number equal to the defending creature's attack value.

Activating 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 turn it will reset itself, if it is a lasting effect, or will stay until the card with the effect is destroyed).

For example, a spell card has an effect that once cast gives a creature an alteration (permanent or temporary) of the stats (attack/defence).

Temporary effect example: an instantaneous spell costs 3 energy, and is a green spell that gives +3/+3 to 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 costs 4 energy and has an effect 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 hands of a player. Effects can influence phases of the game and can modify so the gameplay.

The effects that require a target cannot be activated if the target is not legitimate (e.g.: if a card affects a land but no lands are present, the effect will not occur).

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

Instantaneous cards may affect each other (e.g.: counter an instantaneous spell) and they can chain together. The resolution of the cards' effects 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/).

Gameplay

Preparation:

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

Each turn is divided into phases:

1) [p*reparation*]: 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 work in this phase).
2) [*drawing*] The player gets a card from his deck, and adds it to his hand. If the player cannot get a card, the player will lose.
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 creature's cards in a place that can attack, once attacked the creature cards are marked as used (as the lands). Only creature 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 fewer than 7 cards, nothing happens.

Expansion 1:

THE DAY OF THE ANCIENT MACHINE

Introducing the artefact expansion!

Artefacts are cards that can be played with colourless energy (energy of *any colour* or *no colour*). The colour of an artefact is neutral.

They are permanent and treated as such when in play.

Artefact cards do not have a colour associated, they are colourless.

They can be played only during the owner's turn.

They can be destroyed with appropriate spells (or effects).

They are not creatures and can't attack or be attacked.

Effects from artefacts are always active, **examples** (others can be imagined):

- all creatures cannot defend against attacks and all the damages inflicted in combat are directly assigned as damages to the owner. This effect will remain active as long as the artefact remains in play, and is active for the owner of the artefact and the opponent.
- All creatures of the opponent deal half damage during their owner's turn.
- The opponent will skip his [*drawing*] phase, and destroy this artefact at the beginning of your [*preparation*] phase.

N.B.: *Even though the set introduces some new mechanics, it should be integrated into the main game as transparently as possible (using good OOP design).*

Expansion 2:

Sleight of hand

This expansion introduces new cards and a new modality of playing cards.

The new effect:

The cards with the effect of “sleight of hand” will be played normally (casting) from the player's hand, but they will not be revealed and will be placed on the game board covered (not revealed, placed facedown). This will delay the activation of the effect to specific moments into the game, for example, the "next attack phase”, “the beginning of your next turn” or “the next opponent’s drawing phase”.

Once the card is revealed the effect takes place and the card is discarded.

Example cards (these are just samples):

Name: “Hidden danger” Colour: Red

Cost: 4 red energy, 2 colourless Kind: red spell

Effect: sleight of hand—> reveal this card at the beginning of the next opponent's turn. 4 damages are dealt to all creatures on the board. The opponent will skip the drawing phase. Discard this card after the effect has been completed.

Name: “known game” Colour: colourless

Cost: 4 colourless energy Kind: artefact

Effect: sleight of hand—> reveal this card at the beginning of the next opponent's attack. 4 damages are dealt to all creatures that can attack now on the board. Discard this card after the effect has been completed.

Name: “killer game” Colour: blue

Cost: 2 blue energy Kind: blue spell

Effect: destroy all covered cards on the board.

The effect of “*sleight of hand*” can be triggered at the beginning or end of any phase.

The new colours:

The new expansion also introduces new colours: the **dual-colour** cards. The new colours will be a combination of two of the already existing ones (including older expansions), except for “colourless”. The allowed combinations will be called dual-colour cards” and will count as the colours that are combining them.

For example, brown-green and blue-red are dual-colour cards. Red-colourless counts as a red card, not a dual-colour one.

If an effect targets the cards of a specific colour (e.g.: green), dual-colour targets with that colour in can be targeted as well (e.g.: green-blue, green-white, etc...).

Dual-colour cards will have a cost of different colours of energy.

E.g.: three green and two brown or one red and two blue.

The colour order is irrelevant: green-brown and brown-green are the same.

N.B.: *Even though the set introduces some new mechanics, it should be integrated into the main game as transparently as possible (using good OOP design).*

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

2 users are playing.

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

Each user has some permanent in the game.

Arnold: 4 lands, of which 4 are giving red energy.

Bryce: 5 lands, of which 3 are giving blue energy and 2 red energy. He also has a red 2/2 creature on the board.

Each user has 7 cards in hand and 10 lives.

Turn 1 A.

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

1) [preparation] 2 lands are restored (of 4).
2) [*drawing*] gets a card from the deck.
3) He has 8 cards
4) [*main phase*] Plays 2 lands (one after the other)
5) He gets 6 energy from his lands (4 red), which are now marked as used
6) He plays "Hidden danger" and leaves it on the ground as a covered card (Effect: sleight of hand—> reveal this card at the beginning of the next opponent's turn. 4 damages are dealt to all creatures on the board. The opponent will skip the drawing phase. Discard this card after the effect has been completed).
7) [*ending*] He now has 5 cards and ends the turn.

End situation: 5 cards in hand, 6 used lands on the board, full life.

End situation: 7 cards, 4 lands on the board, full life.

Turn 1 B.

User B (Bryce):

1) [*preparation*] The effect of “Hidden danger” is triggered (this drawing phase is going to be skipped), and the card is discarded into the owner's discard pile.
    1) 0 lands are restored (of 5).
    2) The red 2/2 creature is discarded (4 damages are delivered to it) into the discard pile.
2) No [*drawing*] phase (skipped)
3) He has 7 cards
4) [*main phase*] Plays 1 land
5) He takes 4 energy (1 blue, 3 red) from his lands and those are now marked as used.
6) He plays: “known game” (sleight of hand—> reveal this card at the beginning of the next opponent's attack. 4 damages are dealt to all creatures that can attack now on the board. Discard this card after the effect has been completed).
7) [*ending*] He now has 5 cards and ends the turn.

End situation: 5 cards, 6 lands on the board, 4 lands are marked as used, full life.

End situation: 5 cards in hand, 6 used lands on the board, full life.

Turn 2 A.

User A (Arnold):

1) [*preparation*] 6 lands are restored.
2) [*drawing*] gets a card from the deck.
3) He has 6 cards
4) [*main phase*] Plays 1 land
5) He takes 2 energy from his lands, which are now marked as used

6) He uses 2 energy from his reserve to cast a permanent (a red creature, 2/2).
7) The creature has the effect that removes a random card from the opponent's hand when it comes into play.
    1) User B discards a card randomly.
    2) The card is put in his discards pile.
8) He decides to attack with the creature just summoned.
    1) [*attack*] the attack triggers the "known game" (sleight of hand—> reveal this card at the beginning of the next opponent's attack. 4 damages are dealt to all creatures that can attack now on the board. Discard this card after the effect has been completed).
    2) Arnold interrupts the resolution of the effect by getting 5 energy to cast a red spell that gives a temporary buff to the creature (till the end of the turn it will add +5/+3 to the target creature).
    3) The damages are dealt
    4) The creature concludes the attack and it deals 7 damage to Bryce.
    5) Bryce life goes to 3.
9) [*ending*] Arnold terminates the turn with 3 cards.
    1) The creature goes back to its original stats (2/2).

End situation: 3 cards in hand, 7 used lands on the board, a permanent creature, full life.

*End situati*on: 4 cards, 6 lands on the board, 3 life.

Turn 2 B.

User B (Bryce):

1) [*preparation*] 4 lands restored.
2) [*drawing*] gets a card from the deck.
3) He has 5 cards.
4) [*main phase*] Uses the energy of 2 lands to cast an artefact
    1) The artefact costs 2 colourless energy

2) The artefact has 2 effects:
   1) The artefact effect is: all creatures deal half damage during their owner's turn (damage will be rounded up).
   2) The artefact effect is: the opponent will skip his [*drawing*] phase, destroy this artefact at the beginning of the owner [*preparation*] phase
3) He gets 1 red energy from one of the lands and casts a red spell that deals 3 damages to the creature of Arnold.
4) The 2/2 red creature gets the damage and gets discarded.
5) [*ending*] He ends the turn with no further actions.

End situation: 3 cards in hand, 7 used lands on the board, full life.

End situation: 4 cards in hand, 6 lands on the board, 2 marked as used, 1 artefact on the board, 3 life.

We are not interested in any action outside the *sample script* (the above sequence play sample).

What you have to do...

THERE IS NO NEED FOR A GUI or a CONSOLE GAME.

KEEP IT SIMPLE!

To complete your final assignment, you must finalise the **implementation** (code) and the required **diagrams**, including the new expansion "Sleight of hand".

We want a UML model that shows the essential structure and behaviour of the game. Imagine it to be the baseline documentation to give to a development partner that could implement the game.

We will provide partial code: *if there is already a design pattern implemented, feel free to alter it/extend it or leave it as is (but in this case there should be probably a space for the same design pattern somewhere else to be implemented). Any change should be commented on so we can quickly see what has been altered. Code changed not in a meaningful way (as to contribute to the structure of the program) will not be considered a valid contribution.*

Required Code:

Implement the provided script (sample play from the previous pages) as a game that gets played, each of the moves of each player will be scripted (hard-encoded) and will make use of the data structures (classes and logic) you implemented. This will result in a text-only sequence of printouts of the turns run (similarly to in the provided script). We want to see the moves a player makes, their effects and the resolution of the effects in one big log listing (**printout only**).

We ask you to implement[1] the most appropriate data structure and make an example of a situation described in **turns 1 and 2** of the script in this description.

Use the most convenient design pattern for:

1) Creating and using cards in the new expansions.

2) Creating and maintaining only one existing general board for the game that collects all the states of the elements placed in it.

3) Imagine having to keep updated on the state of each card that comes into play. If your opponent has an effect that affects the card, all the affected cards should be updated accordingly. Create a generalised version that can sustain expansions with dedicated channels of communication for any kind of event.

You must have at least 4 of the following design patterns newly implemented in the right place: Singleton, Factory, Observer, Composite, and State design.

Required diagrams:

Each picture should have inside both the *name* and *student number* of each author. Student1 and student2 are the student numbers. In case of lone delivery student2 is not needed, png/jpg is ok as long as the level of compression allows to read the details).

- We require the **class diagram** of the whole solution.
  - It should contain all the elements implemented in the C# files: the class diagram should represent the implementation of all the entities involved in the required run.
  - Highlight with colours/area selection the design patterns in the diagram (this will make our correction faster).
  - The name of the file should be represented as follows:

[1] See the suggestion section for hints on how to develop better-quality code.

- Ex: CD_student1_student2_INFSAD.jpg

- We require a **finite state machine diagram** for the cards (that are used in your implementation).
  - The name of the file should be represented as follows:
    - Ex: SMD_student1_student2_INFSAD.jpg
- We require the **Sequence diagram** of turn 2.
  - The name of the file should be represented as follows:
    - Ex: SD_student1_student2_INFSAD.jpg

Suggestions!

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: you will have a minimal set of classes that represent the entities but once you start to look at the possible behaviours and application of design principles (for example the design patterns) you will have several other classes. Most of your classes are close to being empty or have just printout codes (console prints).

Suggestion: if the game is in an advanced stage (not in turn one), you can implement a method that "creates 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.

DELIVERY

Groups should be at most 2 students.

Files to deliver:

UML diagrams PICTURES to be submitted

3 files to deliver NO ASTAH or other files, only the pictures:

- CD_student1_student2_INFSAD.jpg
- SMD_student1_student2_INFSAD.jpg
- SD_student1_student2_INFSAD.jpg

C# Files to be submitted

The implementation of all the classes in the class diagram and the run described in any other diagram requested.

The implementation should be compilable with .NET 6.x .

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

Inside each file, there should be the data of each author. Ex:

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

Zip file:

The whole solution with the diagrams should be compressed in a zip file, which is THE ONLY FILE to upload in the form.

Only **zip** (no rar/7zip/others).

The global zip file should be named: INFSAD_student1_student2_2122.zip

It should contain all the files and subdirectories needed.

It must be **cleaned** of any temporary file besides the solution/project one.

Delivery deadline and link:

Deadline: the end of week 9 of this period. **14 April 11:30 PM**.

Link: SPECIFIC LINK TO SUBMIT WILL BE PROVIDED HERE

Evaluation:

To be sufficient:

- Must contain all the elements requested, and each diagram should have the requested depth to understand the structure and the general behaviour from the exercise description. THEY MUST BE USEFUL.
- Must be coherent with the implementation (all the class 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). If there are meaningful differences in usage/implementation and design, the assignment will be graded as insufficient.
- The code must be cross-platform.
- The delivery must be correct in all its parts, the code should be executable correctly, and the diagram should not contain errors in the syntax. A compiler error will result in insufficient by default, regardless of the kind of error.
- The design patterns and structure of the concepts explained in class should be correctly implemented in diagrams and code (using design patterns as a checklist or having them with no concrete 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. Unreadable or unclear solutions will be graded as insufficient.
- No external libraries/installation lib should be needed.
- To run the solution, no input or writing files should be required/ produced.

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

Checklist FOR THE STUDENT:

We would like to avoid reminding the obvious, but experience shows that our obvious is not always as such.

CHECK IF YOUR NAME IS PRESENT ON EACH FILE DELIVERED!

*.cs files: C# files for all the classes.

The solution does not produce errors, and there is no dependency on external libraries.

The solution should contain all the design patterns

The solution should be implementing code avoiding as much as possible hard-encoding of types and any other quick fix that does not consider the game's future expansion.

In the pictures of the diagrams:

check the 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 have the necessary format.

Keep the evaluation criteria at hand and check if you did not do something.

REMEMBER: **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. The perspective of the future expansion should allow you to integrate code without rewriting your already implemented classes.

ATTENTION: If by adding an expansion to the game you are going to have to reimplement classes, “you are doing it wrong”. If you need an enumeration, again—> “you are doing it wrong”.