Project LP2A: One-Hand-Game

Table of contents :

Introduction

Presentation of the game "One Hand" :

"One Hand" is a unique card game that requires a standard 52-card deck. The game involves strategic thinking and luck as players draw and manage a hand of four cards at a time. The objective is to score points by matching specific card attributes and efficiently managing your hand. This game combines elements of solitaire with unique rules that challenge players to think ahead and make tactical decisions.

Project objective :

The objective of this project is to develop a digital version of the "One Hand" game, focusing on designing the game logic, creating an engaging user interface, and ensuring a smooth gameplay experience.

- *Game development:* Implement the rules and mechanics of "One Hand" as a functional and playable software application.
- *User experience:* Develop an intuitive and engaging user interface to make the game easy to understand and enjoyable to play.

This project will allow us to apply the skills and knowledge we have acquired during our LP2A course, translating theoretical concepts into practical software development.

Game rules :

- Setup: Begin with a shuffled 52-card deck.

- Initial Hand: Draw four cards to form your initial hand.

- Scoring:
  - If the first and last cards in your hand have the same number, discard all four cards and score 5 points.
  - If the first and last cards have the same suit, discard the two middle cards and score 2 points.

- Drawing New Cards:
  - If neither condition is met, move the last card behind the third card and draw a new card to be the first card, ensuring you always have four visible cards in your hand.
- Using the Joker:
  - You can discard the two middle cards if no other scoring conditions are met. This can be done three times per game, scoring zero points each time.

Identification of user needs :

The primary user need for the "One Hand" game is an engaging and easy-to-understand gameplay experience. Players will seek a straightforward interface and clear instructions to quickly grasp the game's rules. The game should provide immediate feedback, visual appeal, and smooth performance, ensuring that users can enjoy the game without frustration. Overall, players need a fun and challenging card game that is accessible and enjoyable on various devices.

Game design

Module design :

1. Description of main modules

- ***Card class:***

    - Represents a single playing card.
    - **Attributes :** *suit* (hearts, diamonds, clubs, spades), *rank* (1 to 13), and *imagePath* (path to the card image).
    - **Methods :** *getSuit*(), *getRank*(), *getImagePath*(), and *toString*() (returns a string representation of the card).

- ***Deck class:***

    - Represents a deck of 52 cards.
    - **Attributes :** *cards* (a list of *Card* objects).
    - **Methods :** *loadDeck*() (loads the deck with 52 cards), *shuffleDeck*() (shuffles the deck), *drawCard*() (draws a card from the deck), and *getSize*() (returns the number of remaining cards in the deck).

- ***Hand class:***

    - Represents the player's hand of cards.
    - **Attributes :** *cards* (a list of four *Card* objects).
    - **Methods** : *replaceCard*() (replaces a card in the hand), *getCard*() (retrieves a card from the hand), and *getCards*() (returns the list of cards in the hand).

- ***OneHandGame class:***

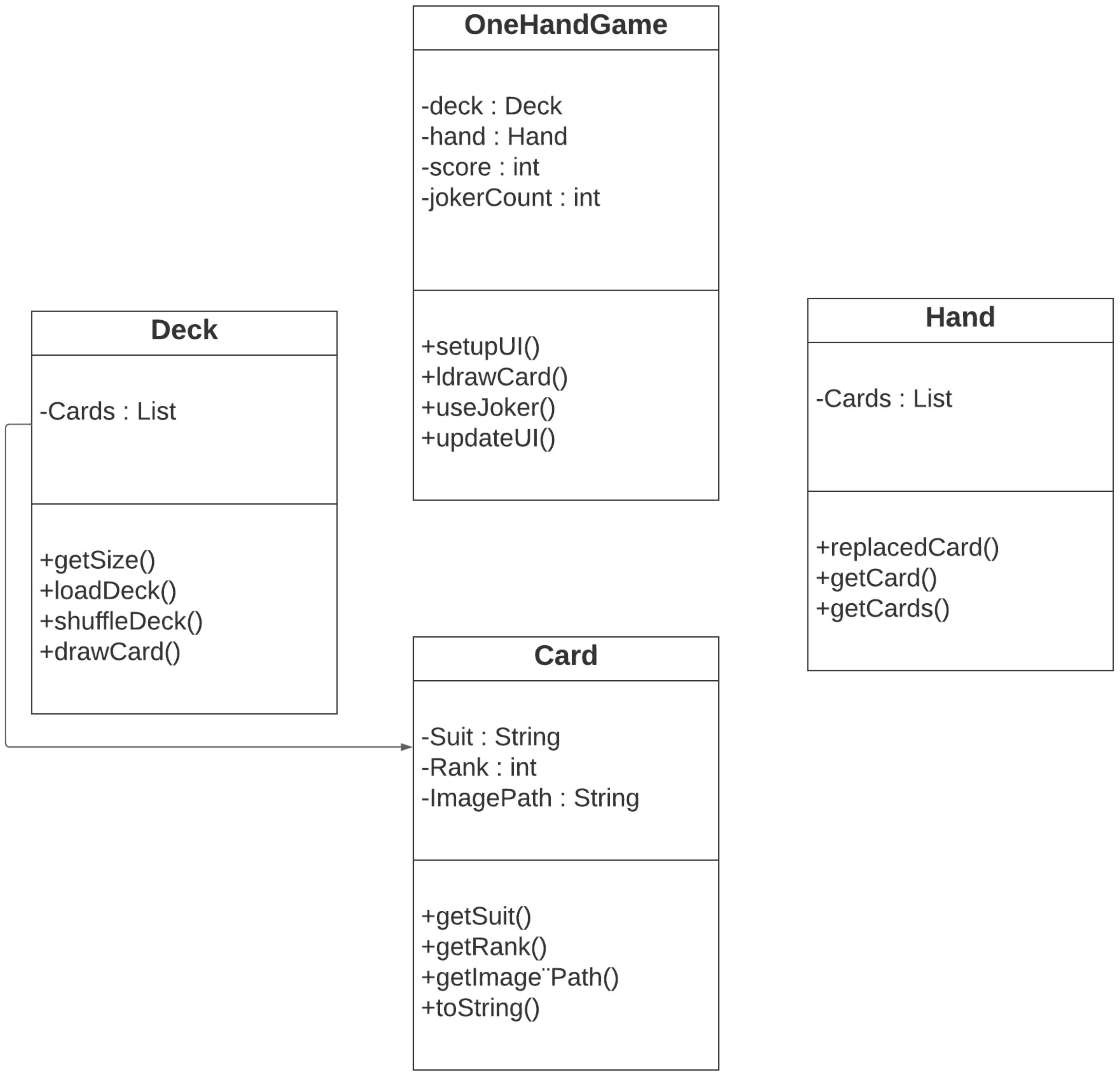
    - Manages the game logic and user interface.
    - **Attributes** : *deck (a Deck* object), *hand* (a *Hand* object), *score* (player's score), *jokerCount* (number of jokers available), *nbOfCards* (number of discarded cards), and *state* (current state of the game).
    - **Methods** : *setupUI*() (sets up the user interface), *drawCard*() (handles the draw action), *useJoker*() (handles the use of a joker), *updateUI*() (updates the user interface), and updateComponentSizes() (adjusts the component sizes based on the window size).

2. Interaction between modules

- The *OneHandGame* class interacts with the *Deck* and *Hand* classes to manage the game flow.
- The *Deck* class provides cards to the *Hand* class.
- The *Hand* class manages the player's hand and interacts with the *OneHandGame* class to update the game state.
- The *Card* class represents individual cards used by the *Deck* and *Hand* classes.

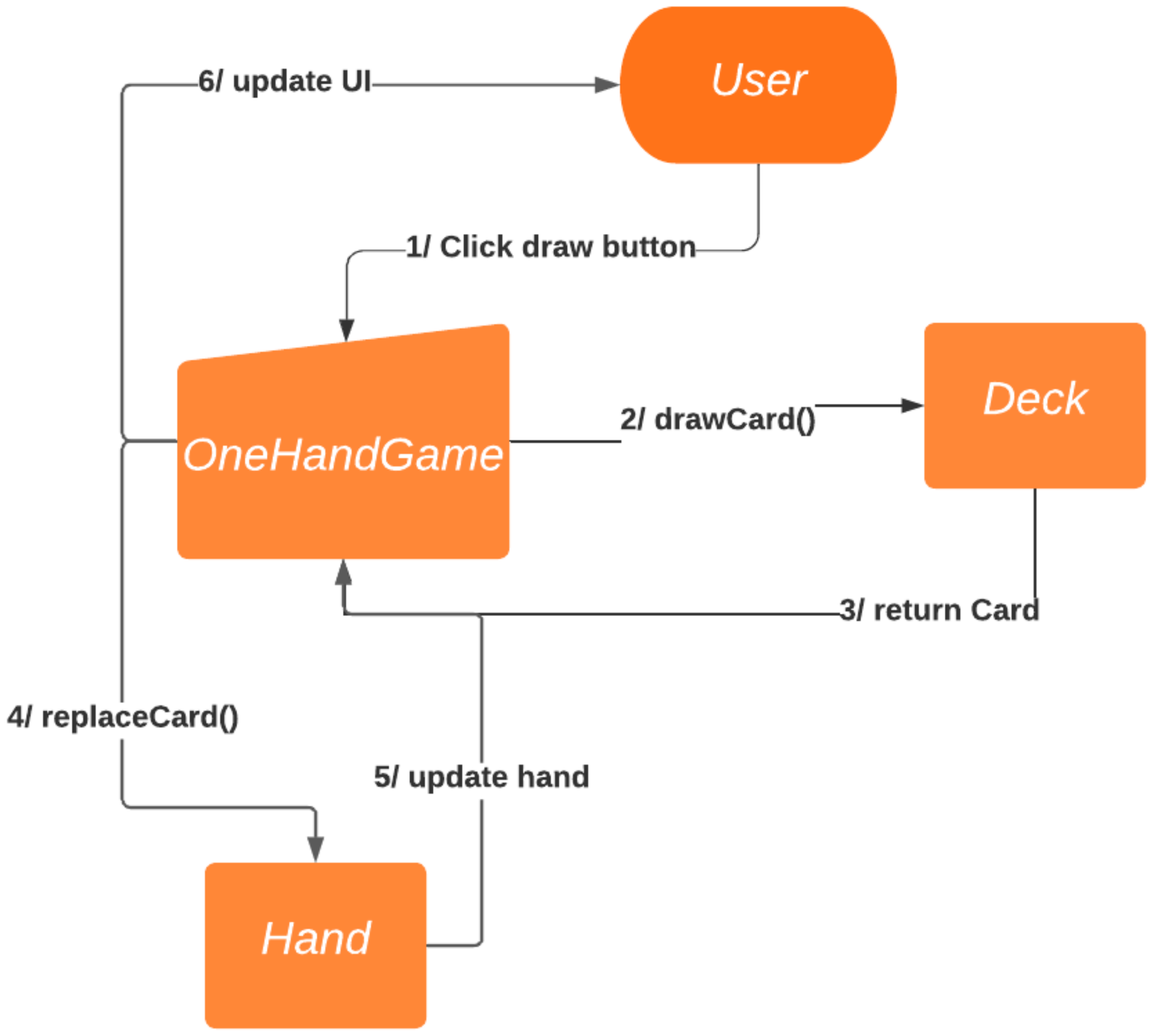
Design Diagrams :

1. Class diagrams



2. Sequence diagrams

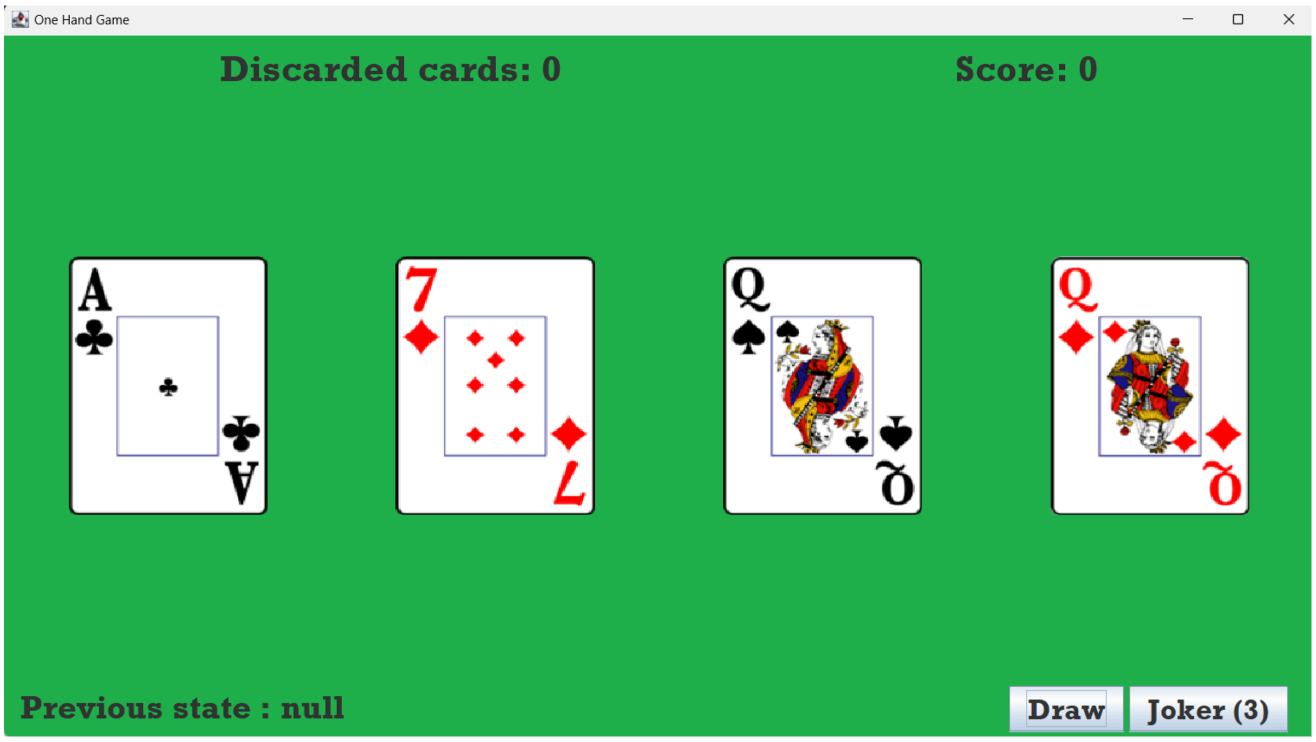
A sequence diagram illustrating the interactions between the *User*, *OneHandGame*, *Deck*, *Hand*, and *Card* classes when drawing a card.



3. User Interface

The user interface for "One Hand" is designed to be intuitive and visually appealing. Key elements include:

Card Display : Shows the four cards in the player's hand.
Score and Discarded Cards : Displays the player's current score and the number of discarded cards.
State Message : Provides feedback on the previous state of the game.
Action Buttons : Includes buttons for drawing a new card and using a joker.
Responsive Design : Adjusts the layout and size of elements based on the window size to ensure a consistent user experience across different screen sizes.



Conclusion

Challenges Encountered and Solutions

We started our work as early as possible to maximize our chances of success. Initially, we focused on coding the game in the terminal to familiarize ourselves with the game mechanics and understand the player's needs. This part was relatively straightforward to develop. However, implementing the graphical interface proved to be significantly more challenging.

Although we initially thought that using form would be very helpful, it turned out to be quite restrictive. Adding images was another difficult aspect to manage, but the most challenging part was implementing the game mechanics within the graphical interface.

Learning Outcomes

Despite these obstacles, we successfully overcame all the challenges and delivered a complete and functional project. This experience not only enhanced our technical skills but also taught us valuable lessons in problem-solving and perseverance.

This project allowed us to become familiar with the Java environment and its specificities. By working on "One Hand," we gained hands-on experience with Object-Oriented Programming (OOP), which opened up new perspectives for us in software development.