Phase 1: Full Development of the Engineering Approach

* Juan Felipe Alarcón G. A00369830
* Juan David Charrupi Silva A00395855
* Appropriate Description of the Problem Context:
  + The problem arises from the need to implement the card game "Uno" in Java, a game of strategy and skill where players try to run out of cards in their hand. The problem context includes the complexity of managing the rules of the game, the interactions between players, and the data structures needed to represent the game state.
* Causes:
  + The complexity of the game "Uno", which involves specific rules for playing cards, drawing cards, and changing the course of the game.
  + The need to manage multiple data structures, such as stacks, queues, hash tables, and priority queues, to represent game state and player cards.
  + The need for a user interface that allows players to interact with the game intuitively and efficiently.
* Symptoms:
  + Difficulty in implementing and maintaining the game logic due to the complexity of the rules.
  + Performance and efficiency issues due to inadequate management of data structures.
  + Lack of clarity in the user interface, making it difficult for players to understand and interact.
* Problem Identification and Definition:
  + The main problem is to efficiently design and implement a card game "Uno" in Java, which complies with the rules of the game and provides a satisfactory game experience for the players. This involves:
  + Implementing the game logic, including card distribution, player turn, move validation, and winner determination.
  + Efficiently managing the data structures needed to represent the game state and player cards.
  + Design an intuitive and attractive user interface that allows players to interact with the game effectively.
* Functional Requirements Specification:
  + Functional requirements associated with the needs stated in the statement include:
    - Distribution of cards to players from a standard 108-card deck.
    - Implementation of game logic to allow players to play cards, draw new cards if necessary, and determine the winner.
* Validation of players' movements to ensure that they comply with the rules of the game.
* Design and implementation of a user interface that allows players to interact with the game effectively.

Phase 2: Evidence of Search Results

During the research, several approaches to the problem of implementing the card game "Uno" in Java have been found, as well as related theoretical and practical elements. Some of the resources used are presented below:

1. Implementation of Data Structures in Java:
   * We studied the slides seen in class for the realization of the different data structures that we had to perform, in addition, we watched some videos on youtube to understand a little clearer some points that we were missing to implement the data structures needed for the game, such as stacks, queues and hash tables.
2. Java Documentation and Tutorials:
   * The official Java documentation, as well as online tutorials, were consulted to understand how to use specific features of the Java language, such as collections and object-oriented programming, in the implementation of the game.
3. Card Game Implementations on GitHub:
   * Repositories on GitHub containing Java card game implementations were explored to get ideas on how to structure the code and manage the game logic.
4. Online Forums and Communities:
   * Online forums and communities, such as Stack Overflow and Reddit, were participated in to raise questions and discuss specific issues related to the implementation of the "Uno" card game in Java.
5. Official UNO Rules and Regulations
   * To have more clarity on how many cards and what the cards are, we looked up the rules of Uno in order to determine how the 108 cards would be distributed.

References:

Java documentation: [Java Documentation - Get Started (oracle.com)](https://docs.oracle.com/en/java/)

GitHub: [stoufa/Uno-Card-Game-GUI-Version: [EN] Uno card game (2D) | [FR] Jeu de cartes Uno (2D) (github.com)](https://github.com/stoufa/Uno-Card-Game-GUI-Version)

Online forums: Stack Overflow, Reddit, etc.

Phase 3: Creative Solution Alternatives

Different ideas were put forward to make the "game" a little more dynamic.

* Autonomous" Game: The option of a "bots" that could play optimally, and that this would generate that only one person could play.
* Implement an efficient card distribution algorithm that guarantees randomness and fairness in the initial distribution.
* Design an optimized move validation algorithm that minimizes processing time and ensures compliance with the rules of the game.
* Design an intuitive user interface with clear and concise messages that guide the player through his turn and available options.
* HasTable: It is considered a good option to store the cards due to its algorithmic complexity of O(1).
* Array: The array is considered to have a list of the current cards.
* Stack: It is considered for the pile of cards to "draw".

Phase 4: Documentation of the Discarding of Non-Viable Ideas

* The ideas of the beginning were discarded since they would add complexity and we would add functions that are not requested in the document, so they were discarded almost from the beginning.
* The option of using the Array was discarded since the stack would be a much more complete option, since that is its function.

Phase 5: Definition of Criteria for Evaluating the Ideas

* Algorithm Efficiency:
  + The time and space complexity of each proposed algorithm will be evaluated, prioritizing those that minimize processing time and system resource usage. This includes the efficiency of the card distribution algorithm and the movement validation algorithm.
* Clarity of User Experience:
  + The clarity and ease of use of the user interface will be evaluated, prioritizing designs that guide the player intuitively through their turn and available options. Intuitive user interface with clear and concise messages will be considered as an important aspect.
* Efficiency of Data Structures:
  + The suitability of proposed data structures, such as Hashtable, Array and Stack, for the storage and manipulation of game cards will be evaluated. Priority will be given to data structures that offer optimal algorithmic complexity for the required operations, such as fast and efficient access to elements.

By evaluating the alternatives against these criteria, the most appropriate solutions for the implementation of the "Uno" game will be determined, prioritizing efficiency, clarity and suitability of the data structures within a game environment guided by user interaction.