**EL MÉTODO DE DISEÑO EN INGENIERÍA**

**CONTEXTO DEL PROBLEMA**

The objective of the project is to develop a card game, specifically "Uno", using the Java programming language and employing various data structures such as stacks, queues, hash tables and priority queues. In this game, players must try to get rid of all their cards by following certain rules. To ensure a systematic and efficient development of the game at all stages, an engineering design approach will be applied.

**FASE 1**

**Necesidad del Sistema:**

The motivation to develop this software comes from the great interest and entertainment generated by the card game "Uno". There is a desire on the part of players to have a digital application that offers the possibility of playing with friends or family, either online or in person, thus preserving the dynamics and competitive spirit characteristic of the traditional game.

**Diseño:**

An engineering methodology will be adopted in the design to ensure that the game implementation is robust and effective. This will include a detailed analysis of the rules of the game, determination of the essential features, and careful selection of the most appropriate data structures and algorithms to carry out the implementation.

**Objetivos Específicos:**

Game details

An exhaustive analysis of the rules of the game "Uno" will be made in order to understand its dynamics, the actions available to the players and how the winner is determined. For example, it will be detailed that the deck is composed of 108 cards, distributed in four different colors (red, yellow, green and blue), each with numbers from 0 to 9, in addition to special cards such as Color Change, Draw 2, Reverse and Jump, along with action cards such as Change of Direction and Jump.

**Estructuras de Datos y Algoritmos:**

Optimal data structures, including stacks, queues, hash tables and priority queues, will be selected to model cards, decks and the current game situation. Also, effective algorithms will be chosen to manage the different in-game actions, such as card drawing and legal move checking.

**Documentacion:**

**Análisis:**

An exploration of the regulations and crucial aspects of the game will be carried out, determining the essential tasks and actions for its proper execution. In addition, emphasis will be placed on the importance of an intuitive user interface that allows players to interact effectively with the game, ensuring a smooth and enjoyable experience.

**Especificación de Requerimientos:**

The necessary elements of the game will be precisely specified, covering guidelines, essential features and the user interface required for effective interaction with the game. Additionally, the importance of help systems and tutorials to guide players through the rules and strategies of the game will be emphasized, thus enhancing their understanding and enjoyment of the game.

**Diseño:**

Data structures and algorithms will be developed for the representation and management of the game state, along with the development of a user interface that facilitates interaction with the game. Additionally, a feedback system will be incorporated to allow users to report errors or suggestions, continuously improving the game experience.

**Implementación:**

The implementation of the essential features of the game will be carried out, ranging from the distribution of cards to the gameplay at each turn, including the handling of special cards and the selection of the winner, all based on data structures and algorithms.

- This approach focusing on the rules, cards and fundamental elements of the game "Uno" creates a robust platform for the development of its Java implementation, ensuring that all vital features of the game are adequately addressed.

**FASE 2: RECOPILACIÓN DE LA INFORMACIÓN NECESARIA**

**TAD:** An Abstract Data Type (ADT) is defined as a specification that details a set of data along with the operations applicable to that data.

Fuente: https://es.wikipedia.org/wiki/Tipo\_de\_dato\_abstracto

**Pila:** A stack is a linear collection of elements that operates under the "Last In First Out" (LIFO) rule. This means that elements are added (push) and removed (pop) from the same end, which is the top of the stack. The essential functions of a stack are therefore the insertion of a new element at the top and the removal of the top element.

*Fuente: https://aprendizdeprogramacion.com/blog/estructuras-de-datos/pilas/*

**Cola:** The queue, a linear type data structure, operates under the FIFO rule, which stands for "First In, First Out". In this structure, new elements are added at the back end and removed at the front end. Among its fundamental operations are enqueue, which consists of adding elements at the end, and dequeue, which involves removing elements from the beginning of the queue.

*Fuente : https://www.ecured.cu/Cola\_%28Estructura\_de\_datos%29*

**Cola de Prioridad:** A priority queue is a data structure that assigns each of its elements a priority level. In this structure, items are processed according to their priority: those with higher priority are processed before those with lower priority. If several items share the same priority, they are handled in the order in which they arrived in the queue.

*fuente :* [*https://barcelonageeks.com/cola-de-prioridad-conjunto-1-introduccion/*](https://barcelonageeks.com/cola-de-prioridad-conjunto-1-introduccion/)

**Generics**: In Java, generics enable the parameterization of types (such as Integer, String and other user-defined types) in methods, classes and interfaces. This facilitates the creation of classes that are compatible with various data types. Thus, a class, interface or method that works with parameterized types is considered generic.

*fuente : https://www.baeldung.com/java-generics*

**Hash Table:** A hash table is a data structure that constitutes an abstract dictionary, where unique keys are associated with specific values. By using a hash function, each key is converted into a unique index within an array, which serves as the location to store the associated value.

*fuente : https://www.udb.edu.sv/udb\_files/recursos\_guias/informatica-ingenieria/programacion-con-estructuras-de-datos/2020/i/guia-8.pdf*

**Juego Uno:** Uno is a card game consisting of 108 pieces and the object is to get rid of all the cards first. During his turn, each player must place a matching card by color, number or symbol with the one on top of the discard pile. If it is not possible to play a card, the player must take one from the deck. There are action cards such as "Color Change", "Draw 2", "Reverse" and "Jump" that alter the dynamics of the game. It is crucial that a player announces "One" when he has only one card remaining.

*fuente : https://www.uno-juego.es/wp-content/uploads/2020/11/UNO-reglas.pdf*

**FASE 3: BÚSQUEDA DE SOLUCIONES CREATIVAS**

The creation of the Uno game will be inspired by the original game, which uses cards and decks. The system will create 76 cards with assigned colors and numbers and 32 action cards, totaling 108 cards. These cards will be organized into queues and the corresponding information will be stored in a Hash Table for reference. Since Uno is a multiplayer game, the system will randomly deal 7 cards from the queue to each registered player. Each player's set of cards will be managed as a stack during the game.

To implement this system, a queue data structure could be used to represent the deck of cards and a stack for each player's deck. The Hash Table would serve to store and quickly access card information during the game.

**FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES**

**1 cola por cada jugador :**

This function will allow the player to review each card individually and select the one he/she wants to play.

* **2 Pilas para el mazo de juego**:

o A pile for the cards not dealt at the beginning of the game.

o Another pile for the cards that have been played and form the discard pile.

* **Cola de robo**:

**A queue for cards that players must draw when they cannot play a card from their hand. This ensures that the card on top of the deck is always drawn.**

**Pila para el mazo de descarte**:

**o** Here are placed the cards that the players have played during their turns.

o The top card of this pile determines which cards can be played by the next players, as they must match in color, number or symbol.

**Cola de prioridad para manejar el orden de los jugadores:**

As players play, their priority is updated and the queue is reorganized to reflect the new order of play.

**FASE 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN**

**Alternative 1:Cola de prioridad para 2 jugadores.**

**Alternative 2:Cola de prioridad para manejar el odern de los jugadores.**

1. **Efficiency.**

Alternative 1: High

Alternative 2:Medium

1. **Flexibility.**

Alternative 1:Medium.

Alternative 2:Low.

1. **Easy Implementation.**

Alternative 1:High

Alternative 2:Medium.

1. **Accuracy:**

Alternative 1:Exact.

Alternative 2:Exact.

1. **Adaptability to real-time changes:**

Alternative 1:Medium.

Alternative 2:High.

1. **Wait time optimization:**

Alternative 1:High.

Alternative 2:Low.

1. **maintenance:**

Alternative1**:**High.

Alternative 2:High.

1. **Fairness in the game:**

Alternative 1:High.

Alternative 2:High.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Criteria** | **Weight** | **Alternative 1** | **Score** | **Alternative** **2** | **Score** |
| **Efficiency** | 20 | High | 5 | Medium | 4 |
| **Implementation** | 20 | High | 5 | Medium | 4 |
| **Maintenance** | 30 | High | 5 | High | 5 |
| **Wait** **time** **Optimization** | 20 | High | 5 | Low | 2 |
| **Adaptability to real time changes** | 10 | Medium | 4 | High | 5 |
| **Total** | 100 |  | 4.8 |  | 4 |

As the final results we choose the altenative 1 as the best option due by his great scores, heres is reasons and how this option is highlighted and implemented in every criteria.

**Efficiency**: The priority queue allows for fast insertion and removal of items. This is crucial in games where the order of players may change frequently and the turn needs to be updated efficiently.

**Flexibility**: Easily adapts to different game rules. For example, if the rules change to give priority to players with fewer points, the priority queue can be quickly reorganized to reflect this new criterion.

**Easy implementation**: Many programming languages already have priority queuing implemented within their standard libraries, making it easy to use without having to implement the data structure from scratch.

**Accuracy**: Priority queuing ensures that the player with the highest priority is always at the front of the queue, ensuring a fair and accurate order of play according to the established rules.

In addition to the advantages mentioned above, priority queuing was chosen for the following additional reasons:

**Adaptability to real-time changes**: In a game, circumstances can change rapidly, and priority queuing can adapt to these changes in real time, reordering players as needed without disrupting the flow of the game.

**Wait time optimization**: By ensuring that players with higher priority play first, waiting time for those further ahead in the game is reduced, which can improve the overall player experience.

**Ease of maintenance**: Because priority queues are a well-known data structure, they are easier for developers to maintain and debug, contributing to a more efficient development cycle.

**Fairness in the game**: Priority queuing ensures that all players have the opportunity to play in an order that reflects their performance or status in the game, which promotes a fair gaming environment.

These features make the priority queue for 2 players a robust and reliable solution for managing the order of players in a game, contributing to a smooth and enjoyable user experience.

**FASE 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES**

Análisis de Complejidad, TAD, Diagrama de clases