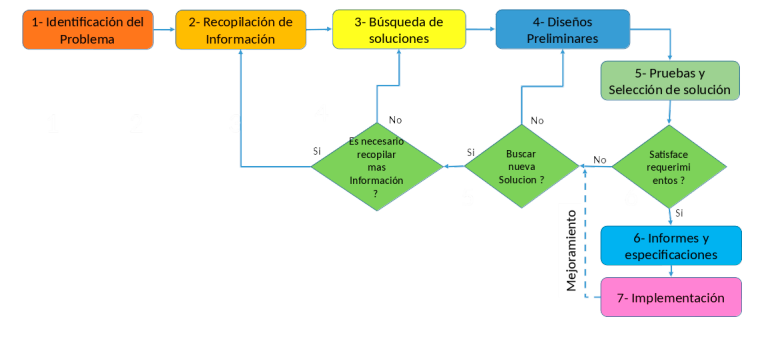
**Problematic context**

A client is in need of implementing a card game in Java that uses a variety of data structures to manage different aspects of the board game known as “Uno”.

**Solution development**

To embark on this challenge , it will be chosen to follow the Engineering Method , a systematic approach and structure for the solutions development , this allows effectively the different aspects of the project.

Based on the description of Engineering Method from the book “Introduction to Engineering” by Paul Wright, the next flow chart has been designed , which will guide the development of the solution step by step:



**Step 1. Identification of the Problem**

-The users want to be able to play “Uno” following their respective rules and game modes.

-It is necessary to implement a system in which various data structures are capable of managing aspects such as the distribution of cards, the players turns and the rules of the game.

-The solution to the problem must be efficient so that the service can be delivered to the greatest number of users with the minimum consumption of resources.

**Step 2. Compilation of information**

An exhaustive search for information related to the data structures and algorithms necessary to implement the card game "Uno" such as hash tables and stacks was carried out through recognized Java pages and online tutorials.

**Sources:**  <https://codegym.cc/es/groups/posts/es.218.tabla-hash-de-java>

<https://www.youtube.com/watch?v=HB5ksj6Op8c>

**Definitions**

Hash table

A Hashtable in Java is a data structure that stores elements in the form of key-value pairs. It works using a "hashing" mechanism, where each key is mapped to an index on a table using a hash function. This allows quick access to the values ​​associated with each key.

Stack

A Stack is a collection of elements that follows the principle of LIFO (Last In, First Out), which means that the last element added to the stack is the first to be removed. In practical terms, a stack can be interpreted as a stack of plates, where the last plate placed on the stack is the first to be removed.

Queue

It is a data structure that follows the principle of "first in, first out" (FIFO). That is, the element that is inserted first into the queue is the first to be deleted. Queues are commonly used to store items in the order in which they are received and processed.

Uno

The Uno card game is a fun, family-friendly board game played with a special deck of cards designed specifically for this game. The main objective of Uno is to get rid of all the cards in the hand before the other players.

Special Cards

The Uno deck includes special cards with additional actions, such as reversing the direction of the game, skipping the next player's turn, having the next player draw additional cards, or changing the playing color. These special cards add strategy and excitement to the game.

TAD

TAD stands for "Abstract Data Type" (ADT). It refers to a programming concept where a data type is defined by its behavior and operations, without revealing the details of its internal implementation. An ADT provides a clear specification of the operations that can be performed on the data, but it does not specify how those operations are implemented. This allows for a clear separation between the public interface (the available operations) and the internal implementation (how those operations are carried out).

**Step 3. Searching creative solutions**

Suggestion of Solution:

•Implement representative classes for the cards and players.

• Design methods for managing stacks, queues, hash tables and priority queues.

• Develop game logic to allow players to play cards and draw new cards according to the "Uno" rules.

•Perform unit and integration tests to ensure correct operation of the system.

**Step 4. Transition from Ideas to Preliminary Designs**

Preliminary design:

Representatives classes:

* • Create classes to represent the cards and players , with relevant attributes and methods.

Management of Data Structures:

* • Implement methods for manipulating stacks, queues, hash tables and priority queues.

Logic of the game:

* •Develop algorithmes to allow players to play cards and draw new cards according to established rules.

Interfaz of Users (UI):

* • Design an intuitive user interface that allows players to interact with the game efficiently.

Tests:

* • Perform extensive system testing to ensure proper operation and compliance with “Uno” rules.

**Step 5. Evaluation and Selection of the Best Solution**

Solution alternatives were evaluated using predefined criteria and the option that best met the project requirements was selected.

Definition of Evaluation Criteria

To evaluate and select the best solution for the Uno card game problem, criteria will be defined based on the needs and requirements identified in the previous phases:

Criterion A. Solution precision:

• [2] Accuracy (precise solution preferred)

• [1] Acceptable approach

Criterion B. Efficiency:

• [4] Constant efficiency

• [3] Efficiency greater than constant

• [2] Logarithmic efficiency

• [1] Linear efficiency

Criterion C. Completeness of functionalities:

• [3] All possible solutions

• [2] Most, but not all, solutions

• [1] Only one solution or none

Criterion D. Ease of algorithmic implementation:

• [2] Compatible with basic arithmetic operations of modern computing equipment

• [1] Not fully compatible with basic arithmetic operations of modern computing equipment

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Solution | **Criterion A** | Criterion B | Criterion C | Criterion D | Total |
| Card Distribution | 2 | 4 | 3 | 2 | 11 |
| Player's Turn | 2 | 3 | 3 | 2 | 10 |
| Rule Verification | 2 | 3 | 3 | 2 | 10 |
| Game Completion | 2 | 3 | 3 | 2 | 10 |
| User Interface (UI) | 2 | 3 | 3 | 2 | 10 |
| Testing | 2 | 3 | 3 | 2 | 10 |

All proposed solutions appear to be viable and obtain a total score of 10 according to the defined criteria. This suggests that each solution has its merits and can be considered based on your projects specific needs and individual priorities.

**Step 6. Preparations of reports and specifications.**

Problem: Implement a system that allows users to play UNO card game.

Tickets:

- play uno game

- Start a new game

- Give the player its current turn

- Give other player the next turn

- Special cards (plus four, plus two, skip turn, change color, reverse card)

Outputs:

* Welcome message and main menu options
* Prompt for number of players and player names
* Message indicating the game is starting and the first player's turn
* Display of the current player's hand
* Prompt for the player to select a card to play
* Message indicating whether the selected card can be played or not
* Message indicating the end of the player's turn and the next player's turn
* Message indicating a player has won the game
* Message indicating a player has run out of cards
* Message indicating a player has played a special card (e.g. +2, +4, skip, reverse)
* Prompt for the next player to select a card to play
* Message indicating the game is over and the winner is announced
* Error messages for invalid input or actions
* Message indicating a player has been penalized for not saying "UNO" before playing their second-to-last card
* Message indicating a player has chosen a new color for the game
* Message indicating the direction of play has been reversed
* Message indicating the current player is skipping their turn
* Message indicating the number of cards in the draw pile and discard pile
* Message indicating the game is paused and the current state of the game
* Message indicating a saved game is being loaded.

Preliminary design:

