

Proyecto grupal final

Luis Fernando Solano Montoya

Julio A. López Álvarez

Universidad Cenfotec

Estructura de datos I

Christian Sibaja Fernández

Tabla de Contenido

[Capítulo 1- Descripción del problema 3](#_Toc79947630)

[Capítulo 2 – Diagramas y TAD 4](#_Toc79947631)

[UML 4](#_Toc79947632)

[Lista 4](#_Toc79947633)

[Cola 7](#_Toc79947634)

[Capítulo 3 - Estado del aplicativo. 9](#_Toc79947635)

[Capítulo 4 - Funcionamiento de la aplicación 10](#_Toc79947636)

[Menú de inicio: 10](#_Toc79947637)

[Opción - definir jugadores: 10](#_Toc79947638)

[Opción - iniciar juego: 11](#_Toc79947639)

[Opción - Mostrar puntaje 12](#_Toc79947640)

[Opción – Mostrar récords ordenados: 13](#_Toc79947641)

# Capítulo 1- Descripción del problema

Se nos ha contratado para crear una versión de un juego para computadora en el lenguaje C++.Se solicita realizar un análisis de los diferentes tipos de juegos y ver cuál de ellos puede cumplir los siguientes requisitos de juego:

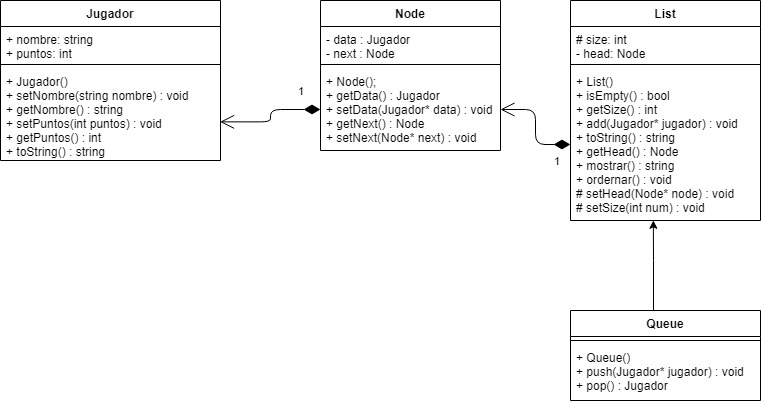
1. Que permita al menos a dos jugadores interactuar por turnos
2. Que, si se hacen varias partidas consecutivas, se pueda guardar el registro de los mejores jugadores (récords), según el tipo de juego (en archivos o base de datos).
3. Debe contener acciones o información tal que permita trabajar con listas, colas, pilas (debe indicar dónde se utilizan). Lo importante es que ojalá estas estructuras sean parte integral de la lógica del programa y puedan darle soporte al juego interactuando entre sí.
4. Se debe programar en capas y bajo el paradigma orientado a objetos.
5. No es indispensable que sea un juego existente (Solitario, UNO, Gran Banco, etc.), se puede hacer una versión personalizada de alguno o inclusive a partir de una idea crear, pero es importante que sea funcional y permita ganar al menos a un jugador (mínimo 2 jugadores, máximo N).

Es por lo que para resolver este problema se decidió realizar el tradicional juego de piedra papel o tijera, donde hay una cola de jugadores a participar en el juego. Las reglas de negocio de este juego son:

* Piedra la gana a tijera.
* Tijera le gana a papel.
* Papel le gana piedra.

# Capítulo 2 – Diagramas y TAD

## UML



## Lista

1. Representación abstracta

Table, Excel

Description automatically generated

* 1. Debe contener como atributos un nodo y un tamaño (size).
  2. Debe ser capaz de obtener y establecer sus atributos, validar si esta vacía, imprimir sus atributos en un método ToString() o mostrar y podrá ordenar sus nodos.

1. Invariante
   1. El nodo es de la clase Node, que contiene la información del jugador y puntos.
   2. Size será de tipo Integrer, que contiene la cantidad de nodos que tiene.
2. Operaciones

List: (nada) -> Lista (Constructora)

// Construye un nuevo objecto List

isEmpty: nada -> booleano (Analizadora)

// Devuelve si la lista esta vacía

getSize: nada -> entero (Analizadora)

// Obtiene la cantidad de elementos de la lista

add: Jugador -> nada (Modificadora)

// Agrega un jugador a la lista

toString: nada -> string (Impresora)

// Obtiene la lista de elementos para guardar en txt

getHead: nada -> Node (Analizadora)

// Obtiene la cabeza de la lista

mostrar: nada -> string (Impresora)

// Muestra la lista de elementos para el usuario

ordenar: nada -> nada (Modificadora)

// Ordena los elementos de mayor a menor

setHead: Node -> nada (Modificadora)

// Modifica la cabeza de la lista

setSize: entero -> nada (Modificadora)

// Modifica el tamaño de la lista

1. Manejo del error
   1. Los errores se controlan en interfaz de UI pues solo se puede ejecutar las funciones desarrolladas por completo, por lo que en TAD de List no se hará ninguna validación.

## Cola

1. Representación abstracta

Table

Description automatically generated

* 1. Esta clase implementa la clase List por lo que sus atributos son los mismos.
  2. Debe ser capaz de atender y agregar nodos.

1. Invariante
   1. El nodo es de la clase Node, que contiene la información del jugador y puntos.
   2. Size será de tipo Integrer, que contiene la cantidad de nodos que tiene.
2. Operaciones

Queue: (nada) -> Queue (Constructora)

// Construye un nuevo objecto Queue

push: Jugador -> nada (Modificadora)

// Agrega a la cola un elemento

pop: nada -> Jugador (Analizadora)

// Elimina el primer elemento de la cola

1. Manejo del error
   1. Los errores se controlan en interfaz de UI pues solo se puede ejecutar las funciones desarrolladas por completo, por lo que en TAD de List no se hará ninguna validación.

# Capítulo 3 - Estado del aplicativo.

La aplicación es completamente funcional, cuenta con un menú que permite a los jugadores completar cualquiera de las siguientes acciones:

1. Definir jugadores
2. Iniciar juego
3. Mostrar puntaje
4. Mostrar récords ordenados
5. Salir

# Capítulo 4 - Funcionamiento de la aplicación

A continuación, se muestra el funcionamiento de cada una de las funciones descritas en el capitulo 3. Al ser un aplicativo de consola, es precisamente esto lo que se mostrara, impresiones de la consola en ejecución.

## Menú de inicio:

Texto

Descripción generada automáticamente

## Opción - definir jugadores:

Texto

Descripción generada automáticamente con confianza media

## Opción - iniciar juego:

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamenteForma

Descripción generada automáticamente con confianza media

## Opción - Mostrar puntaje

Texto

Descripción generada automáticamente

## Opción – Mostrar récords ordenados:

Texto

Descripción generada automáticamente