## I. Contextualización Breve del Problema

El proyecto consiste en la implementación del juego de estrategia Othello en C++, siguiendo un enfoque de programación orientada a objetos (POO). Othello es un juego de tablero para dos jugadores que se juega en un tablero de 8x8, con fichas de dos colores (negro y blanco). El objetivo del juego es tener la mayor cantidad de fichas de tu color en el tablero al final de la partida. Las reglas del juego incluyen la colocación de fichas, la captura de fichas del oponente y el seguimiento de los puntajes. Además, se debe registrar el historial de las partidas.

## II. Análisis

### 1. Consideraciones para el Abordaje del Problema

Para abordar la implementación de Othello, se han tomado en cuenta las siguientes consideraciones:

* Modularidad: Se ha diseñado el proyecto de manera modular, utilizando clases para representar las distintas entidades del juego, como el tablero, los jugadores y el administrador del juego. Esto facilita la organización del código y permite una fácil extensión en el futuro.
* Interfaz de usuario simple: La interacción con el juego se realiza a través de la consola, con una representación amigable del tablero utilizando asteriscos '\*' para fichas blancas y guiones '-' para fichas negras.
* Registros de partidas: Se ha implementado un sistema de registro de resultados de partidas en un archivo de texto, que almacena el nombre de los jugadores, la fecha, la hora, el ganador y la cantidad de fichas al final de cada partida.
* Uso de excepciones: Se utilizan excepciones para manejar posibles errores, como movimientos inválidos o problemas en la lectura/escritura de archivos.

### 2. Desarrollo de Propuesta de Estrategia de Solución

La estrategia de solución se basa en la creación de clases que representan las entidades del juego y su interacción. A continuación, se detallan las clases principales:

* Player: Representa a un jugador con un nombre, un color (negro o blanco) y un puntaje. Esta clase permite gestionar la información de cada jugador de manera independiente.
* Board: Modela el tablero de juego y almacena la disposición actual de las fichas. Contiene métodos para validar movimientos, realizar capturas y mostrar el tablero en la consola.
* Game: Administra el flujo del juego, incluyendo los turnos de los jugadores, la verificación del fin del juego y la determinación del ganador.
* HistoryManager: Se encarga de registrar y almacenar los resultados de las partidas en un archivo de texto, así como de imprimir el historial de partidas.

## III. Diseño

### Diagrama de Clases

### Texto Descripción generada automáticamente con confianza media Justificación de Decisiones

* **Clase Player**: Se utiliza para representar a los jugadores, manteniendo su nombre, color y puntaje. Esto permite un seguimiento individualizado de la información de cada jugador.
* **Clase Board**: Representa el tablero de juego. Contiene la lógica para validar movimientos, realizar capturas y mostrar el tablero en la consola. El tablero se representa mediante una matriz bidimensional de objetos BoardSquare, lo que facilita el seguimiento de cada casilla.
* **Clase Game**: Administra el juego, incluyendo los turnos de los jugadores y la lógica del juego. Mantiene una referencia al tablero y a los jugadores actuales, lo que permite un control centralizado del flujo del juego.
* **Clase HistoryManager**: Se encarga de la gestión de los registros de partidas en un archivo de texto. Esto proporciona una forma de mantener un historial de partidas jugadas.
* **Clase BoardSquare**: Esta clase no aparece en el diagrama de clases, pero se utiliza para representar cada casilla del tablero. Contiene información sobre el color de la casilla y si está vacía o no.