**Logotipo

Descripción generada automáticamente**

**Informe del Parcial 2**

Primera entrega

Presentado por:

Juan Sebastian Pérez Rubiano

Presentado A:

Aníbal José

Augusto Salazar

Departamento de Ingeniería Electrónica y Telecomunicaciones

Universidad de Antioquia

Medellín-Antioquia

Octubre de 2023

**Índice**

1. Contextualización breve del problema.
2. Análisis:

consideraciones para el abordaje del problema y desarrollo de propuesta de la estrategia de solución.

1. Diseño:

un diagrama donde refleje y describa las clases que conforman su solución. Incluya los miembros de clase y el especificador de acceso en cada caso. Además, aborde la lógica detrás de los principales algoritmos implementados en los métodos y funciones desarrolladas. Justifique sus decisiones.

1. Algoritmos implementados:

anexe un enlace al repositorio con los archivos fuente. e. Experiencia de aprendizaje: Problemas de desarrollo que afrontó, evolución de la solución, consideraciones y resultado del aprendizaje alcanzado durante el desarrollo.

1. Contextualización:

se trata de la implementación de un juego de estrategia llamado Othello; es un juego de mesa de estrategia para dos jugadores. El objetivo del juego es tener más fichas de tu color en el tablero al final de la partida.

1. Análisis:

Al ver el juego, me di cuenta de que se podía representar inicialmente mediante una matriz del tamaño del tablero. Sin embargo, tras leer los requisitos del trabajo, me percaté de que no se permitía usar vectores. Por lo tanto, decidí utilizar un array de dos dimensiones para mantener las posiciones de filas y columnas. Inicialicé este array con espacios en blanco y luego agregué el carácter de la ficha en las posiciones iniciales. Para gestionar un juego por turnos, creé un bucle.

Después de abordar la creación y visualización del tablero, me enfoqué en cómo abordar el problema del encierro tipo "sándwich". La verificación de la dirección ingresada por el jugador resultó ser sencilla, ya que solo era necesario verificar el tamaño y asegurarse de que la posición no estuviera ocupada. Sin embargo, al abordar el encierro tipo "sándwich", me di cuenta de que debía examinar las ocho direcciones posibles: arriba, derecha, izquierda, abajo y sus diagonales (derecha-arriba, izquierda-arriba, izquierda-abajo). Para abordar esto, implementé un bucle que recorría cada dirección. Luego, desarrollé un método para dirigirme en cada dirección específica, lo que implicó identificar un patrón que me permitiera recorrer cada dirección mientras las fichas fueran diferentes a las del jugador (es decir, las del oponente) y luego verificar si la siguiente ficha pertenecía al jugador.

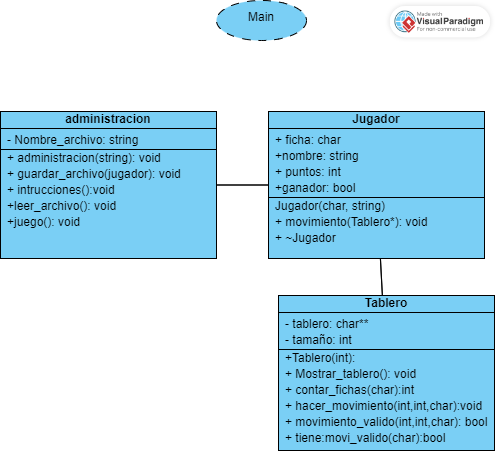
Para realizar movimientos, simplemente actualicé el array. Usé un enfoque similar al utilizado para verificar los movimientos, con la diferencia de que, si en una dirección había un movimiento válido, reemplazaba las fichas del oponente por las del jugador.

Para determinar si un jugador tenía movimientos válidos, implementé un bucle que recorría cada posición de la matriz y verificaba los movimientos disponibles. También creé una clase que almacenaba el nombre y la ficha de cada jugador. Para llevar un registro de los puntos, agregué un atributo a la clase del jugador.

La determinación del ganador se basó en comparar los puntos de ambos jugadores. Además, consideré que el juego podría finalizar antes de llenar todo el tablero si ninguno de los jugadores tenía movimientos válidos. En ese caso, se verificaba quién había ganado o si se producía un empate.

Finalmente, para llevar un registro de las partidas, planteé la necesidad de agregar funciones de lectura y escritura en una base de datos que almacenara la fecha, hora, jugadores, puntos y ganador de cada partida. Para guardar, simplemente obtuve la hora del sistema y registré los demás datos de los jugadores en una sola línea en el documento. En cuanto a la lectura, procesé los datos línea por línea para mostrar la tabla.

Diseño:



4 repositorio:

<https://github.com/sebastianrubiano/Parcial-2.git>

5 video: