UNIVERSIDAD DE SANTIAGO DE CHILE

FACULTAD DE INGENIERÍA

Departamento de Ingeniería Informática

PROYECTO DE LABORATORIO 3 – PARADIGMA ORIENTADO A OBJETOS

Paradigmas de Programación

Gerardo Ignacio Pérez Encina 21.343.296-6

Sección:

13204-0-A-1

Profesor:

Edmundo Leiva Lobos

Fecha:

26 de diciembre del 2024

TABLA DE CONTENIDO

1. INTRODUCCIÓN

Conocer diferentes paradigmas de programación es esencial para que los desarrolladores puedan abordar problemas de manera estructurada y eficiente, seleccionando las técnicas más adecuadas para cada contexto. Este enfoque permite construir soluciones que no solo son funcionales, sino también sostenibles y escalables.

En el ámbito del desarrollo cognitivo y educativo, los juegos como Conecta 4 destacan por fomentar habilidades clave como el razonamiento lógico, la planificación estratégica y la resolución de problemas. Este laboratorio aborda la implementación de Conecta 4 en el lenguaje Java, utilizando el paradigma de programación orientado a objetos (POO), que facilita la modelación del juego como un sistema de entidades interactivas.

El documento incluye la introducción, descripción del problema, análisis de requisitos, diseño e implementación de la solución, instrucciones de uso, resultados y conclusiones. Este enfoque busca demostrar cómo el paradigma POO es una herramienta poderosa para resolver problemas complejos y, al mismo tiempo, resaltar cómo este juego puede contribuir al desarrollo de competencias cognitivas de una manera interactiva y entretenida.

2. DESCRIPCIÓN DE LA PROBLEMÁTICA

2.1. PROBLEMA

Cuando investigamos más sobre el juego de Conecta 4, nos damos cuentas que nuestro problema es la implementación de este, pero primero hablemos sobre que es Conecta 4, este juego trata sobre un tablero de 6x7 el cual se juega con dos jugadores, cada jugador tiene una cantidad de piezas y en cada turno solo pueden colocar una pieza, la cual cae desde arriba hasta el fondo o hasta donde sea posible, para poder determinar un ganador se debe de hacer una línea vertical, horizontal o diagonal.

2.2. DESCRIPCIÓN DEL PARADGIMA

La programación orientada a objetos (POO) es un paradigma que organiza el código en torno a objetos que representan entidades del mundo real o conceptual, encapsulando datos y comportamientos dentro de clases. Este enfoque facilita la reutilización, la escalabilidad y el mantenimiento del código.

A continuación, se describen los conceptos clave del paradigma orientado a objetos aplicados a este proyecto:

- **Clases y objetos:** Una clase es una plantilla que define atributos (datos) y métodos (comportamientos). Los objetos son instancias de una clase que interactúan entre sí para resolver problemas. Por ejemplo, en este proyecto, las clases Player, Board y Game representan los elementos principales del juego, y sus instancias corresponden a los jugadores y al estado del tablero.

- **Encapsulación:** Consiste en ocultar los detalles internos de las clases y exponer solo lo necesario a través de métodos públicos. Esto asegura que los datos sean manipulados únicamente mediante interfaces bien definidas, lo que reduce errores. Por ejemplo, el acceso y modificación de las estadísticas de un jugador (Player) se realiza mediante métodos como getWins() y setWins().
- **Abstracción:** Permite enfocarse en los aspectos relevantes de un objeto, ignorando detalles innecesarios. Por ejemplo, en el caso de la clase Piece, solo se considera el atributo color para identificar a qué jugador pertenece la ficha, sin modelar otros detalles irrelevantes para el juego.
- **Herencia y Polimorfismo:** Aunque no se usaron directamente en este proyecto, son fundamentales en POO. La herencia permite que una clase derive de otra, reutilizando y extendiendo funcionalidades. El polimorfismo permite que un mismo método tenga comportamientos diferentes según la clase que lo implemente, como podría ser un método genérico para validar jugadas en diferentes variantes del juego.

3. ANÁLISIS DEL PROBLEMA

Para solucionar el problema, se desea crear el juego Conecta 4 en el lenguaje de programación Java, bajo el paradigma de programación orientado a objetos. Esto quiere decir que se debe crear un programa que, mediante funciones, se consiga poder jugar una partida de Conecta 4 sin problemas. Por consiguiente, se debe de desarrollar una solución en la cual podamos de realizar el juego Conecta 4 sin ningún problema.

El Conecta 4 está constituido por: Jugadores, tablero y piezas. Y como requisitos, el programa de debe cumplir con las siguientes funcionalidades: Creación, gestión de la partida, gestión de jugadores (Registrar nuevo jugador, modificar información de algún jugador existente, consultar estadísticas de los jugadores), desarrollo del juego (Realizar los movimientos, validar los movimientos según las reglas del juego, detectar victoria, empate o continuación del juego, mostrar el estado actual del tablero).

Para la construcción del programa, se hizo un análisis previo de un diagrama de clases UML (ver Anexo 7), donde se establece la implementación del código en POO, los atributos y métodos de las clases, junto a los tipos de relaciones entre estos. Por ejemplo, en una primera instancia se determina que el objeto Player tiene como atributos una ID (int), un nombre (String), un color (String), wins (Int), losses(Int), draws(Int) y remainingPieces(Int), junto a sus respectivos métodos como el método constructor, los getters y setters.

La creación de estos tipos de datos abstractos no conlleva grandes dificultades si se siguen los pasos del análisis previo. Aun así, la definición de los métodos de los TDAs requiere un mayor trabajo. En el anexo 2 se presenta una tabla con estos métodos y la explicación de ellas.

4. DISEÑO DE LA SOLUCIÓN

Para desarrollar una solución que implemente el juego Conecta 4 en Java, es esencial descomponer el problema en sus componentes principales y considerar las reglas del juego. El paradigma de programación orientado a objetos (POO) resulta ideal para modelar los elementos del juego como entidades independientes que interactúan entre sí.

TDA Player:

Es el jugador que uno define al crear la partida, este tiene una id, nombre, color, victorias, derrotas, empates y cantidad de fichas, a lo largo que se vaya jugando, sus últimas cuatro características se irán actualizando conforme vaya pasando la partida.

TDA Board:

Es el tablero del juego, la base de este, se podrá crear un tablero, el jugador podrá poner fichas en el tablero y verificar el estado de la partida (Si se ganó, perdió, o hay un empate).

TDA Piece:

Es la pieza que el jugador seleccionará, esta puede ser del color la cual el jugador desee.

TDA Game:

Este es el “TDA madre” el cual tendrá a las demás para poder conformar el juego, este servirá para poder actualizar la partida, es decir, nos mostrara como se ponen las fichas en el tablero, verificar de quien es el turno y etc., también ver el historial y poder visualizar el tablero.

A medida que se iba creando el código, surgieron ciertos cambios con respecto al diagrama de análisis inicial (Ver Anexo 8). Uno de los cambios mas notables es la eliminación de los setters que poseía Player.

Uno de los algoritmos más importantes es la detección de victorias es uno de los aspectos más cruciales del juego Conecta 4. Para lograrlo, el algoritmo recorre el tablero verificando si existen líneas de 4 fichas consecutivas en las direcciones vertical, horizontal o diagonal. Este proceso implica evaluar cada celda del tablero y analizar las posiciones adyacentes en las tres direcciones mencionadas. Si se encuentra una coincidencia consecutiva de fichas del mismo color en cualquiera de estas direcciones, el algoritmo determina que el jugador correspondiente ha ganado la partida. Este enfoque garantiza una evaluación precisa del estado del juego en cada turno.

5. CONSIDERACIONES DE IMPLEMENTACIÓN

5.1. ESTRUCTURA DEL PROYECTO

El proyecto se desarrolló utilizando el paradigma orientado a objetos en el lenguaje Java. Este enfoque permitió la encapsulación de los diferentes componentes del juego en clases específicas, cada una con sus propias responsabilidades.

Clases principales:

1. **Game:** Es el núcleo del programa, actúa como controlador principal que gestiona las interacciones entre el tablero (Board) y los jugadores (Player).
2. **Board:** Representa el tablero del juego. Se encarga de almacenar el estado actual del tablero y proporciona métodos para colocar piezas y verificar victorias o empates.
3. **Player:** Modela a los jugadores del juego, almacenando su información personal, estadísticas y fichas restantes.
4. **Piece:** Representa las fichas de los jugadores, asociando cada ficha con un color específico.

5.2. INTERPRETE USADO

- Lenguaje utilizado: Java (versión 17).
- Entorno de desarrollo integrado (IDE): IntelliJ IDEA, seleccionado por sus herramientas avanzadas para depuración, gestión de proyectos y facilidad de integración con Gradle.
- Compilador: Java SE Development Kit (JDK) 17.
- Sistema de construcción: Gradle, utilizado para gestionar dependencias y compilar el proyecto.

6. INSTRUCCIONES DE USO

Para comenzar a jugar con el programa siga estos pasos (Windows 11).

1.- Navegue a la carpeta que contiene los archivos del proyecto (Anexo 3)

2.- Abra una terminal en esa ubicación. Esto puede hacerse haciendo click derecho dentro de la carpeta y seleccionando “Abrir en terminal” (Anexo 4)

3.- En la terminal, ejecute el siguiente comando “.\gradlew.bat build” sin las comillas (Anexo 5)

4.- En la misma terminal, ejecute el comando “.\gradlew.bat run”. (Anexo 6)

Windows 10

1.- Navegue a la carpeta que contiene los archivos del proyecto (Anexo 3)

2.- Abra una terminal en esa ubicación. Esto puede hacerse apretando Shift y Click derecho dentro de la carpeta y seleccionando “Abrir la ventana de PowerShell aquí” (Anexo 5)

3.- En la terminal, ejecute el siguiente comando “.\gradlew.bat build” sin las comillas (Anexo 5)

4.- En la misma terminal, ejecute el comando “.\gradlew.bat run”

Una vez completado el paso anterior se mostrará por pantalla el programa. Asegúrese de seguir las instrucciones que se muestran en pantalla.

7. RESULTADOS Y AUTOEVALUACIÓN

Los resultados correspondientes a las RF se detallan en el archivo "autoevaluación_21343296_GerardoPerezEncina". Sin embargo, se puede afirmar que el programa funciona correctamente hasta el RF 19, siempre y cuando se respeten las instrucciones de uso.

Habiendo dicho eso, el proyecto ha demostrado ser exitoso en la implementación del juego Conecta 4. El código cumple con los objetivos planteados, logrando así una simulación completa del juego.

8. CONCLUSIONES

Este proyecto representó un desafío técnico al implementar Conecta 4 en Java utilizando programación orientada a objetos. El enfoque permitió estructurar el sistema de manera modular y escalable, destacando la importancia de la planificación y el diseño temprano. La encapsulación y la separación de responsabilidades facilitaron la implementación de funciones clave como detección de victorias, validación de movimientos y gestión de turnos, garantizando la funcionalidad del sistema.

El paradigma POO demostró ser eficaz para manejar la complejidad del proyecto, aunque requirió un diseño cuidadoso para evitar acoplamientos y garantizar la claridad en la interacción entre componentes. El proyecto cumplió con los objetivos establecidos, pero deja espacio para mejoras como la integración de interfaces gráficas y pruebas automatizadas para mayor robustez.

En resumen, el desarrollo reafirmó la importancia de aplicar principios de ingeniería como modularidad, reutilización y diseño claro, logrando una solución funcional y extensible que sienta las bases para futuros proyectos más complejos.

9. REFERENCIAS

Kuhn, T. S. (1962). *La estructura de las revoluciones científicas.*

E, S. (2024, 19 septiembre). Connect 4 for Cognitive Development: How the Game Boosts Brain Power. Elakai Outdoor.

Spigariol, L. (2005). *Fundamentos teóricos de los Paradigmas de Programación.* Buenos Aires:

10. ANEXO

1.

Nombre TDA	Dato abstracto	Atributos	Operaciones relevantes
Board	Representación del tablero, donde se colocarán las fichas	Filas, Columnas, tablero	Crear Board, canPlay, placePiece, verticalWin, horizontalWin, diagonalWin, entregarGanador
Player	Representación del jugador junto a sus atributos	Id, name, color, wins, losses, draws, remainingPieces	Crear Player, actualizarStats
Piece	Representación de la pieza para jugar sobre el tablero	color	Crear Piece
Game	Representación del juego Conecta 4, formandose a partir de las TDAs mencionadas anteriormente	Board, player1, player2, currentPlayer, historial, colorPlayer1, colorPlayer2	Crear Game, showHistory, esEmpate, getCurrentPlayer, boardGetState, endgame, realizarMovimiento

Tabla 1: Especificación de los TDAs usados

2.

Metodo	Procedimientos
canPlay	Verifica si es que aun se puede jugar en el tablero
playPiece	Coloca una ficha en la columna especificada, siendo esta en la posición mas baja
verticalWin	Verifica si hay una secuencia de piezas del mismo color de manera vertical
horizontalWin	Verifica si hay una secuencia de piezas del mismo color de manera horizontal
diagonalWin	Verifica si hay una secuencia de piezas del mismo color de manera diagonal
entregarGanador	Determina si hay un ganador verificando todas las posibles condiciones de victoria (Vertical, horizontal o diagonal)
showHistory	Muestra el historial de los movimientos realizados
esEmpate	Verifica si el juego ha terminado en empate (Ya sea porque no tiene mas casillas o porque los jugadores se quedaron sin piezas)
actualizarEstadisticas	Actualizas las estadisticas wins, losses o draws dependiendo del caso
getCurrentPlayer	Obtiene el jugador actual
boardGetState	Muestra el estado del tablero actual
endGame	Finaliza el juego actualizando las estadísticas de los jugadores y mostrando el resultado
realizarMovimiento	Realiza el movimiento de un jugador, verifica el estado del juego y gestiona los turnos

Tabla 2: Explicación de procedimientos de los requerimientos funcionales

3.

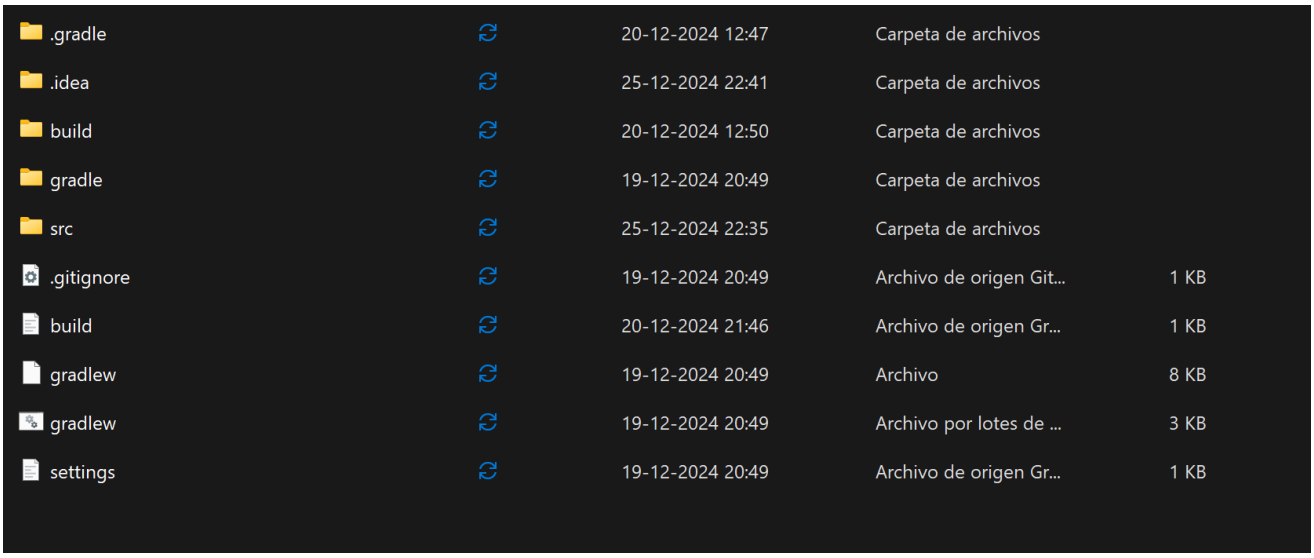

Ilustración 1: Carpeta fuente del programa

4.

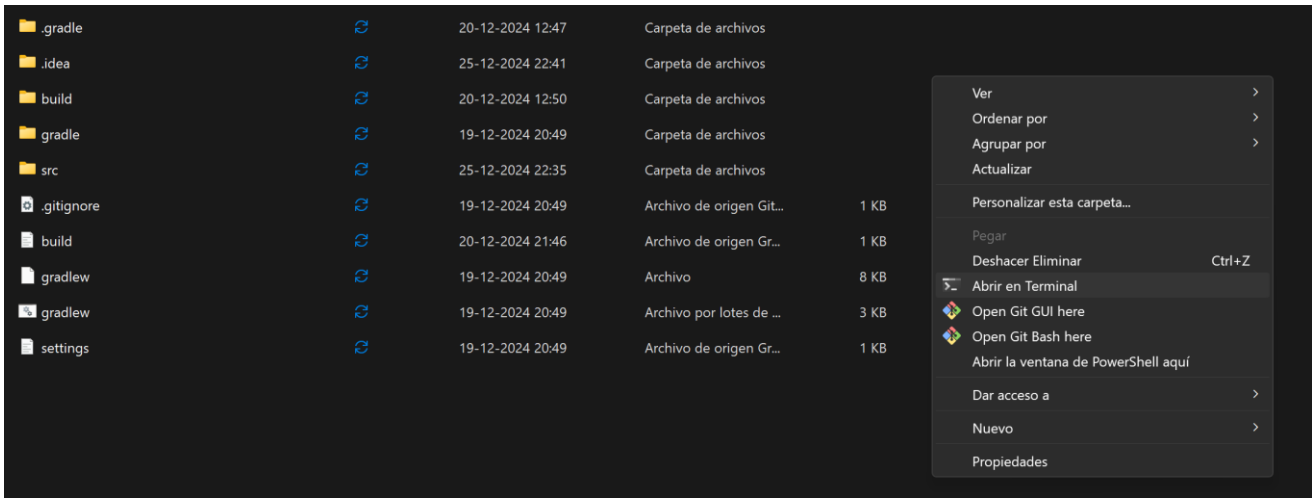

Ilustración 2: Abrir carpeta fuente en terminal

5.

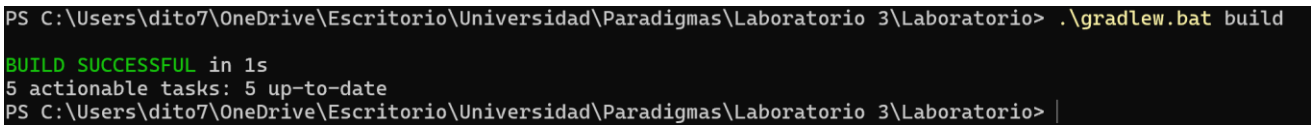

Ilustración 3: Código ".\gradlew.bat build" en la terminal

6.

```
PS C:\Users\dito7\OneDrive\Escritorio\Universidad\Paradigmas\Laboratorio 3\Laboratorio> .\gradlew.bat run

> Task :run
-----MENU PRINCIPAL-----
Bienvenido a Conecta 4

Seleccione una opcion
1.- Crear nuevo juego
2.- Visualizar estado del tablero
3.- Realizar jugada
4.- Ver estadisticas
5.- Salir del juego
Ingrese su opcion:
<==========----> 75% EXECUTING [3s]
> :run
```

Ilustración 4: Código ".\gradlew.bat run" en la terminal

7.

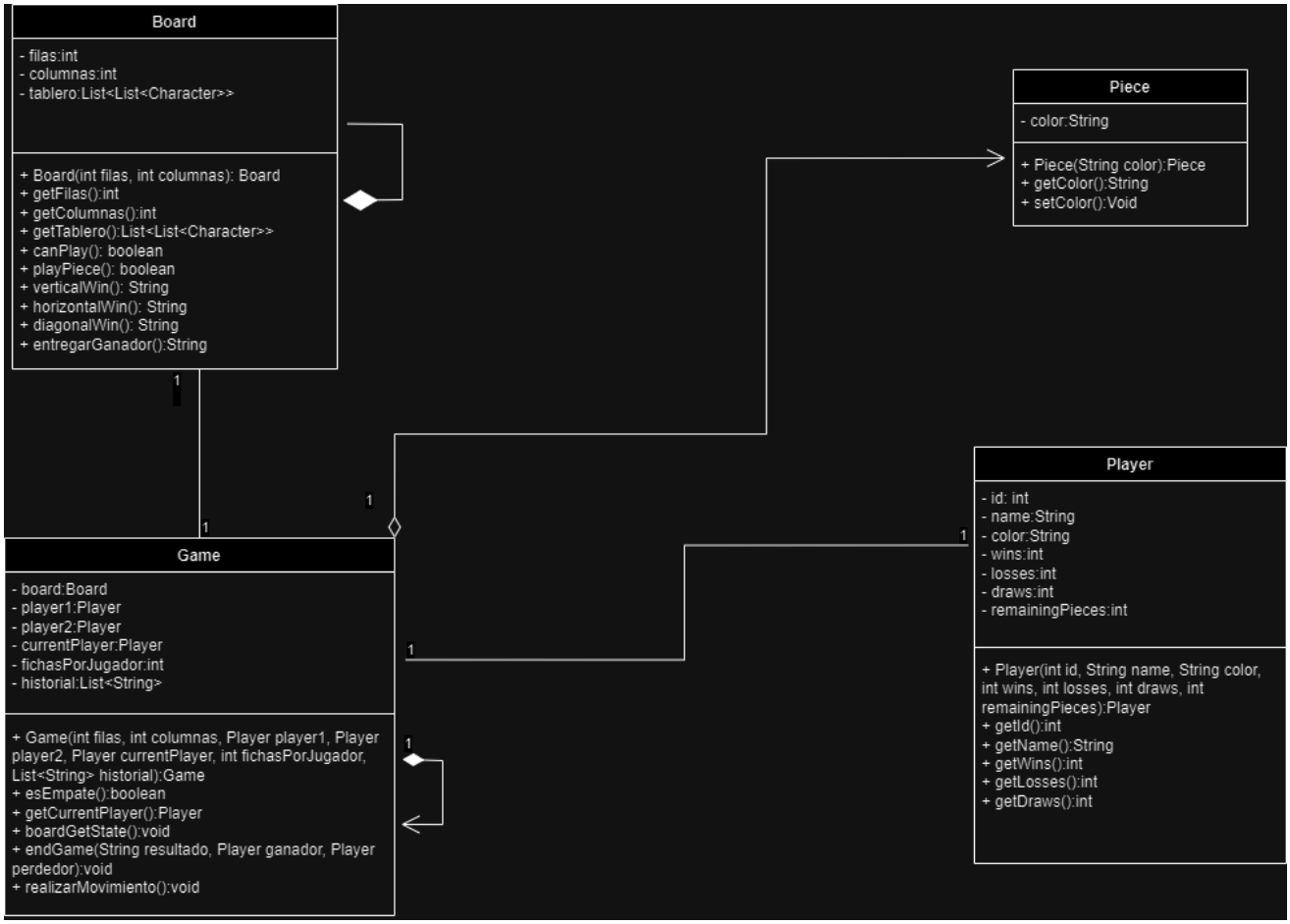

Ilustración 5: Diagrama de clases UML de análisis

8.

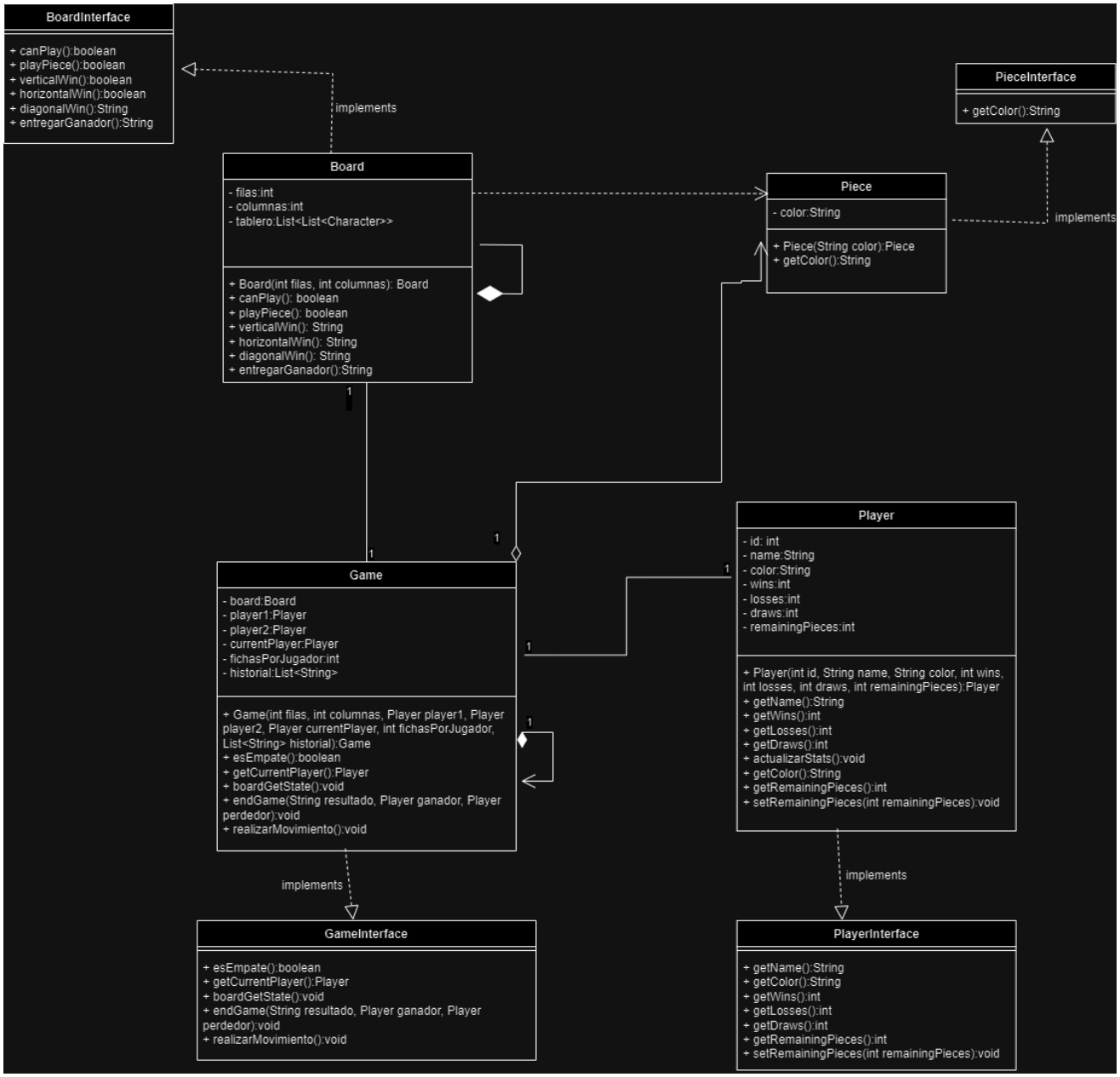

Ilustración 6: Diagrama de clases UML posterior al desarrollo