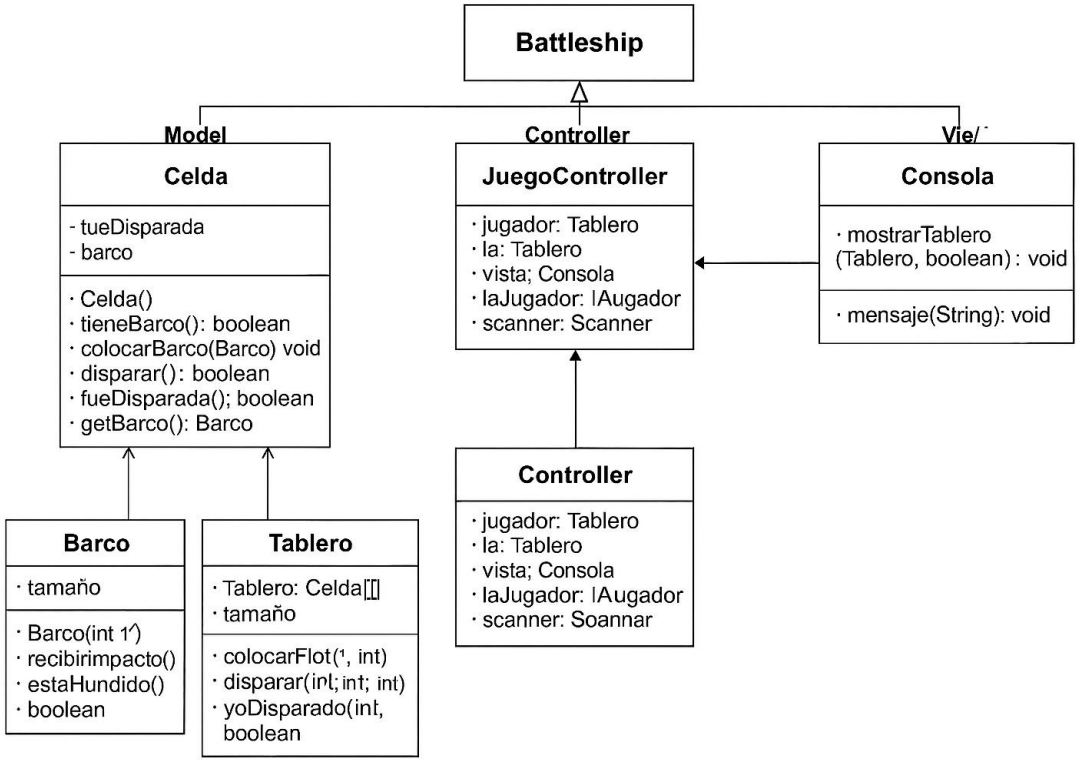

README

Este proyecto implementa el juego de Battleship en Java utilizando el patrón de arquitectura Modelo-Vista-Controlador (MVC). Permite jugar contra una IA básica que dispara aleatoriamente.

Justificación del diseño y uso de estructuras en el proyecto Battleship (MVC)

1. Uso de matriz de objetos:

Se utiliza una matriz de objetos `Celda[][]` en la clase `Tablero` para representar el tablero de juego. Cada celda es un objeto que encapsula información sobre si contiene un barco, si ha sido disparada y qué barco contiene. Esta estructura permite una representación clara y flexible del estado del juego, facilitando el acceso y modificación de cada posición del tablero. El uso de objetos en lugar de tipos primitivos permite extender fácilmente la funcionalidad (por ejemplo, agregar efectos visuales, estados adicionales, etc.).

2. Decisiones de diseño para respetar MVC:

El patrón Modelo-Vista-Controlador se implementa separando claramente las responsabilidades:

- Modelo: Incluye las clases `Barco`, `Celda` y `Tablero`, que contienen la lógica del juego y el estado interno.

- Vista: La clase `Consola` se encarga exclusivamente de mostrar información al usuario y recibir entradas, sin acceder directamente al modelo.

- Controlador: La clase `JuegoController` coordina la interacción entre la vista y el modelo, gestionando los turnos, validaciones y flujo del juego.

Esta separación permite mantener el código organizado, facilita la realización de pruebas unitarias en el modelo sin depender de la interfaz, y permite cambiar la vista (por ejemplo, de consola a gráfica) sin modificar la lógica del juego.

Además, se incluye una clase `IAJugador` desacoplada que permite simular un oponente automático, lo que demuestra la extensibilidad del diseño.

Este enfoque modular y desacoplado mejora la mantenibilidad del código, facilita la colaboración en equipo y permite futuras mejoras sin afectar el núcleo del sistema.