Documentación del Controlador y Modelos del Juego Naval

Tabla de Contenidos

ControladorPartida

Clase Singleton que gestiona la lógica principal de la partida. Se encarga de instanciarse una única vez y de iniciar nuevas partidas entre jugadores.

Los métodos `crearPartida` utilizan los objetos `IPuntuacion` e `IMovimiento` como plantillas. Estas interfaces permiten duplicar su estado mediante los métodos `clonePuntuacion()` y `cloneMovimiento()`, garantizando que los datos originales no se alteren durante la ejecución de la partida.

Métodos relevantes

- `getInstance(Scanner scaner)`: Inicializa y devuelve la instancia del controlador usando un Scanner para entrada del usuario.
- `getInstance()`: Devuelve la instancia existente. Lanza una excepción si no ha sido inicializada.
- `crearPartida(IJugador, IJugador, IPuntuacion, IMovimiento)`: Crea una partida entre dos jugadores ya definidos.
- `crearPartida(IJugador, IJugador, IPuntuacion, IMovimiento, boolean espera)`: Crea una partida entre dos jugadores con control sobre la espera entre turnos.
- `crearPartida(IJugador, IPuntuacion, IMovimiento)`: Solicita el nivel de dificultad y crea una partida con IA.

Puntuación

- Se calcula según los impactos ('I'), hundimientos ('H') y fallos ('A'):

    - I: +2 puntos
    - H: +5 puntos
    - A: -1 punto
- 20 puntos extra para el ganador.

Interfaz IJugador

Interfaz que representa a un jugador dentro del juego. Define los métodos esenciales que deben implementar tanto jugadores humanos como IA.

Métodos

- `boolean aceptarAccionComplementaria(TBarcoAccionComplementaria tipo, int cantidadDisponible)`
  Determina si el jugador puede realizar una acción complementaria especial, como lanzar artillería, reparar, etc., en función del tipo de acción y cuántas veces queda disponible.

- `int[] realizaTurno(char[][] tablero)`
  Ejecuta un turno del jugador. Devuelve un array con la coordenada de ataque (fila, columna) sobre el tablero enemigo.

- `void addMovimiento(IMovimiento movIn)`
  Añade un objeto de tipo IMovimiento al historial de movimientos del jugador.

- `String getNombre()`
  Devuelve el nombre del jugador.

- `void addPuntuacion(IPuntuacion punIn)`
  Añade un objeto de tipo IPuntuacion al historial del jugador.

- `List<IMovimiento> getMovimientos()`
  Devuelve la lista de todos los movimientos realizados por el jugador.

- `List<IPuntuacion> getPuntuaciones()`
  Devuelve la lista de puntuaciones acumuladas por el jugador.

Interfaz IPuntuacion

Representa una puntuación en el sistema.

Métodos

- `getPuntos()`: devuelve los puntos obtenidos.

- `clonePuntuacion()`: clona el objeto puntuación.
- `setPuntuacion(long)`: asigna puntuación.
- `setPartidaId(Long)`: asigna el ID de la partida asociada.

Interfaz IMovimiento

Representa un movimiento realizado durante la partida.

Métodos

- `cloneMovimiento()`: clona el movimiento actual con datos vacíos.
- `setPartidaId(Long)`: vincula el movimiento a una partida.
- `setFila(int)`, `setColumna(int)`: establece la posición.
- `setTime(long)`: asigna el instante temporal.

Interfaz Algoritmo

Define el comportamiento básico de un algoritmo de ataque de la IA.

Métodos

- `getNextAttack(char[][] tableroOpuesto)`: devuelve la siguiente posición a atacar.

Observaciones

- Las acciones especiales están unificadas mediante el enum `TBarcoAccionComplementaria`, en lugar de cadenas.
- La estructura permite extensibilidad fácil para nuevas estrategias, tipos de jugador, y modos de visualización.

Fin del documento.