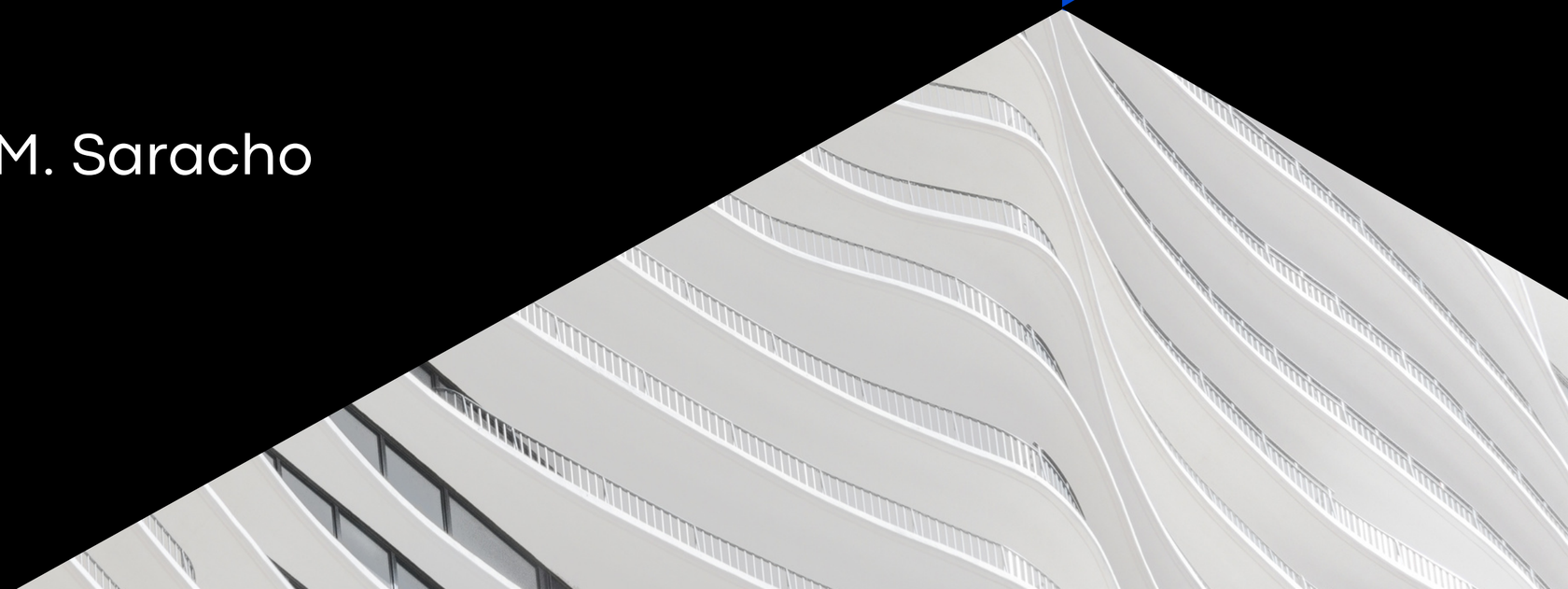
23/02/2024 - FINAL DE PROGRAMACIÓN ORIENTADA A OBJETOS

Juego: El Molino

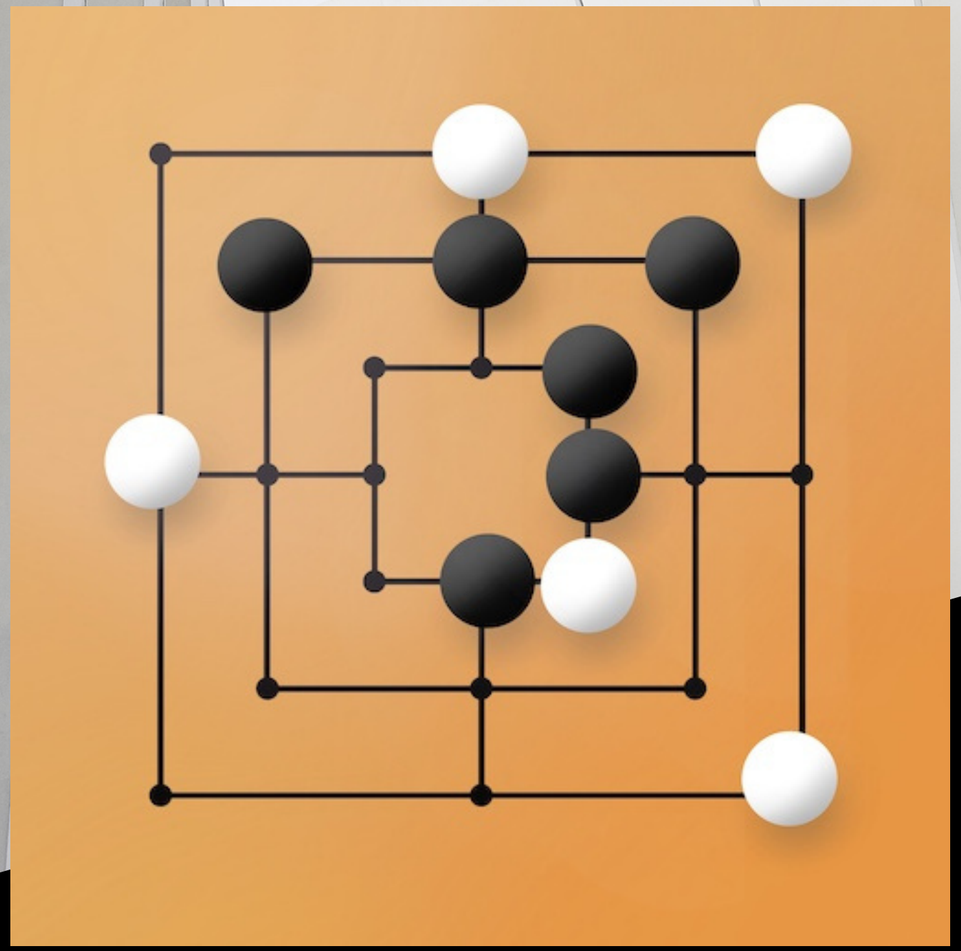
Nueve hombres de Morris o
Alquerque de Nueve

Presentado por Franco M. Saracho
Legajo: 182783



Temario

- ¿Qué es El Molino?
- ¿Cómo se juega? Reglas
- Elementos que componen al juego
- Muestra de los elementos en código
- Patrón MVC y Observer
- Interfaces visuales implementadas
- ¿Cómo funciona el juego en red? y persistencia de datos



¿Qué es el Molino?

Primero conozcamos un poco sobre de qué va este juego de mesa.

¿Qué es el molino? ¿En qué conciste?

El juego del molino es un juego de mesa de estrategia abstracto de mesa para dos jugadores originado en el Imperio romano.

El juego es mencionado como alquerque de nueve en el Libro de los juegos y también es conocido como nueve hombres de Morris o Morris.

Primero los jugadores entran en la fase de colocar 9 fichas. Luego comienzan a moverlas. Llegado a este punto, los jugadores se centraran en hacer “molinos”, combinaciones de 3 fichas, sea horizonal o verticalmente.

El jugador que se quede con 2 fichas o sin movimientos, pierde la partida.

¿Cómo se juega? Reglas y estados del juego

Los jugadores se turnan para colocar sus piezas en las intersecciones vacías. Si un jugador es capaz de formar una fila de tres piezas a lo largo de una de las líneas del tablero, tiene un "molino" y puede eliminar una de las piezas de su oponente en el tablero; las piezas retiradas no se pueden volver a poner en juego.

Una vez que las 18 piezas se han colocado, los jugadores se turnan moviendo.

Para mover, el jugador desliza una de sus piezas a lo largo de una línea en el tablero a una intersección vacía adyacente. Si no puede hacerlo, ha perdido el juego.

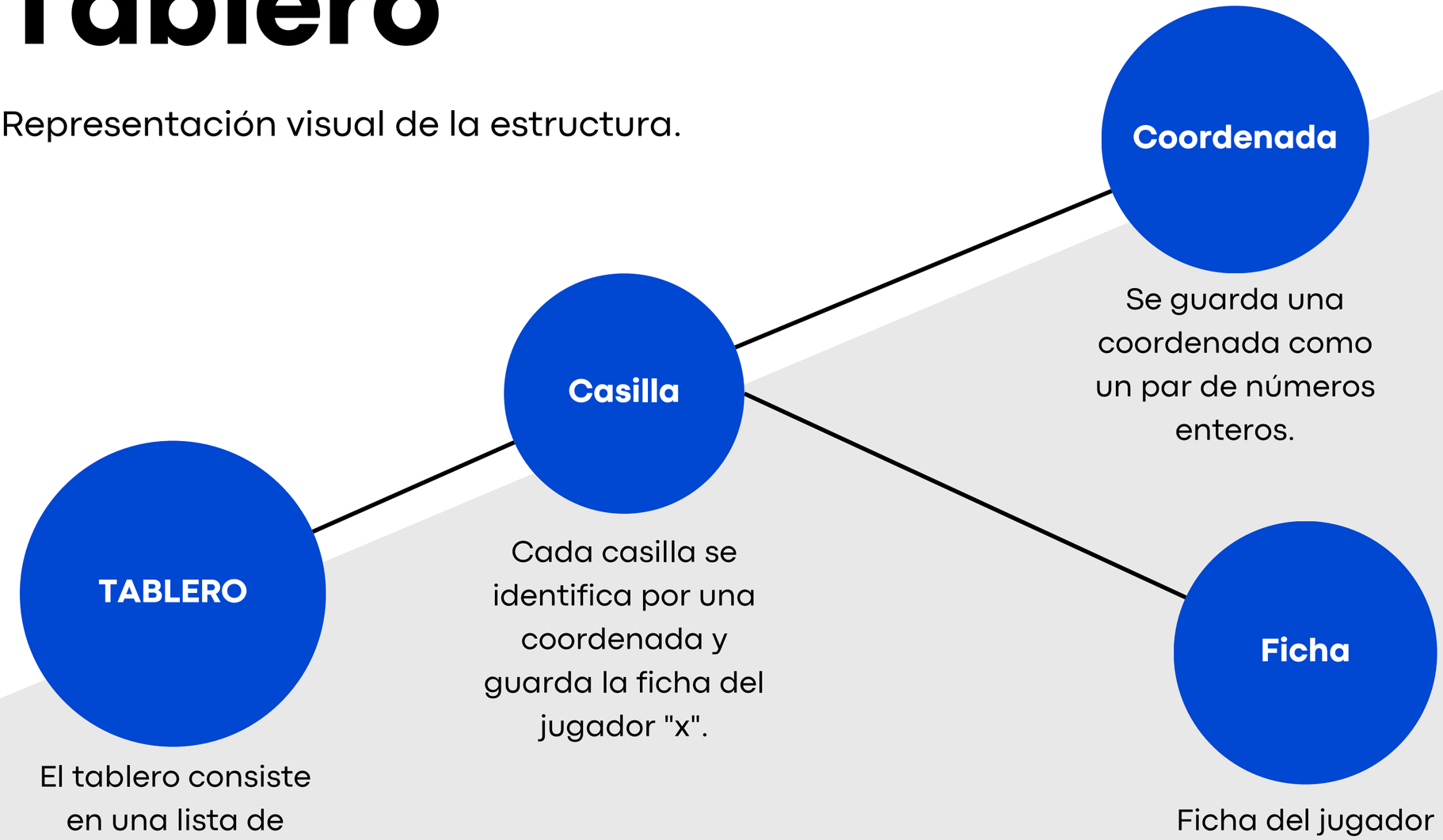
En una variante común, una vez que un jugador es reducido a tres piezas, sus piezas pueden "volar", "brincar" o "saltar" a cualquier intersección vacía, no solo a las adyacentes.

Elementos que componen al juego

TABLERO	COORDENADA
CASILLA	FICHA
JUGADOR	

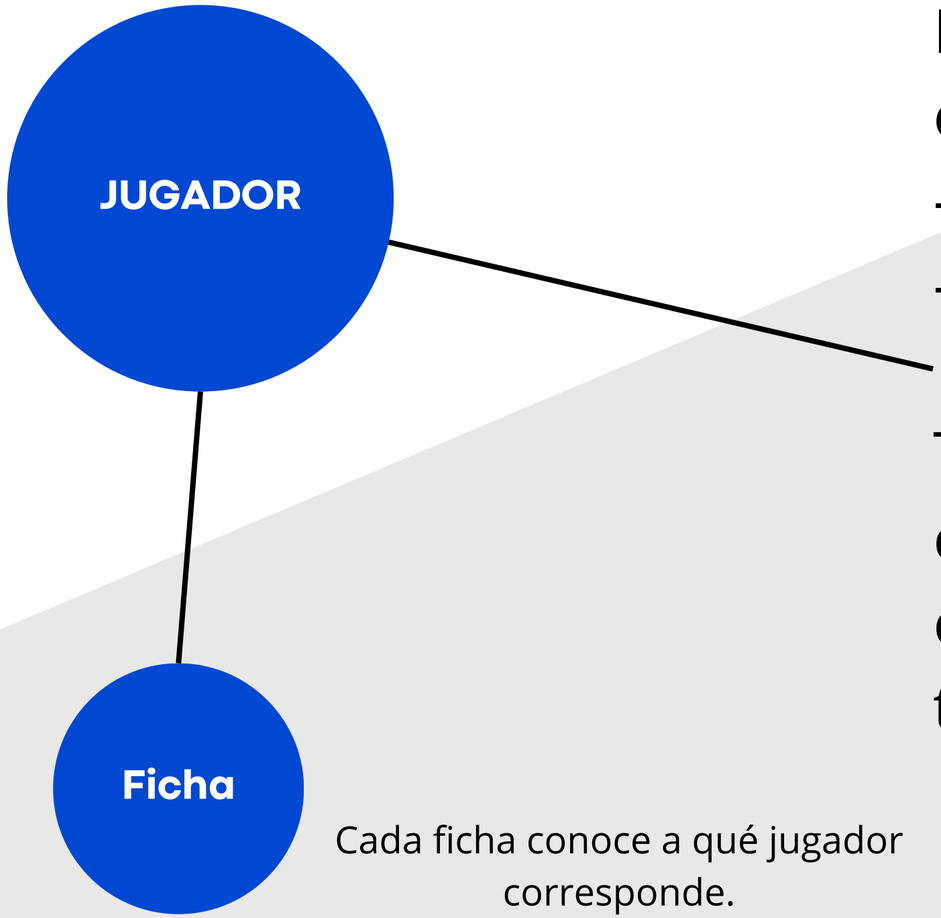
Tablero

Representación visual de la estructura.



Jugador

Representación visual de la estructura.

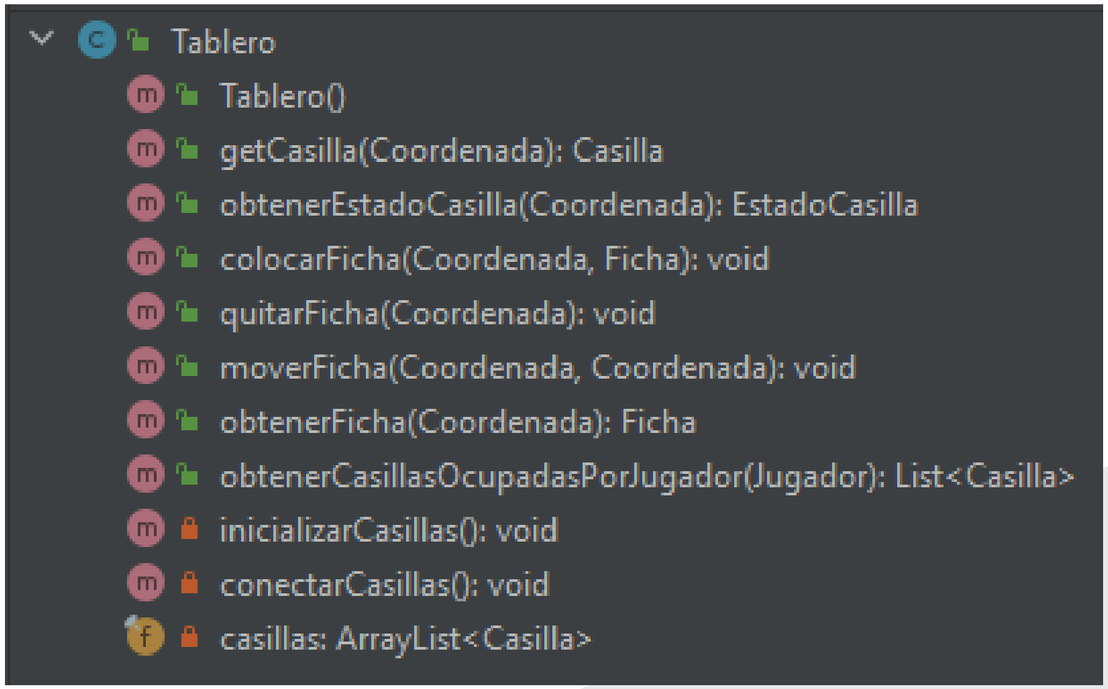


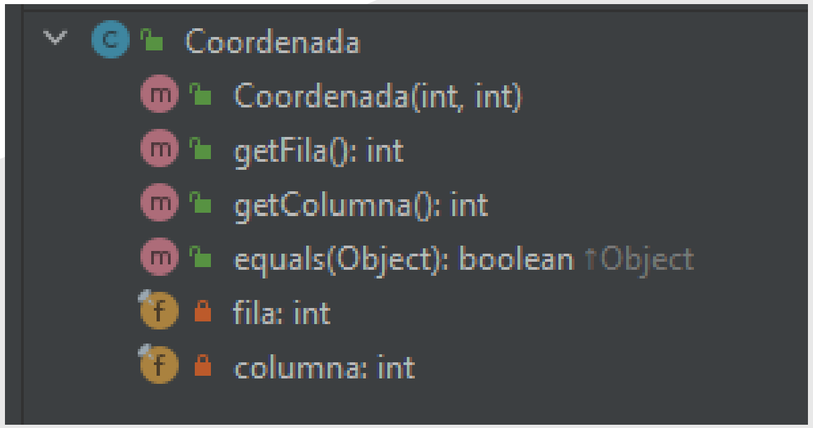
El jugador está compuesto por una estructura básica:
-- ID
-- NOMBRE

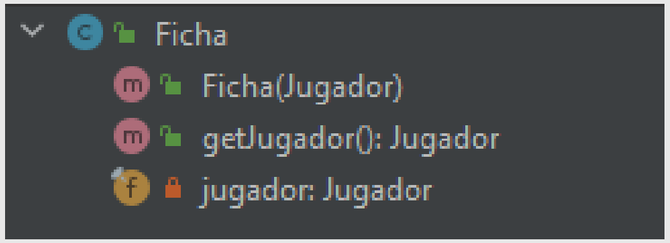
También, acá se lleva registro de cuántas fichas se colocaron y cuántas hay actualmente en el tablero.

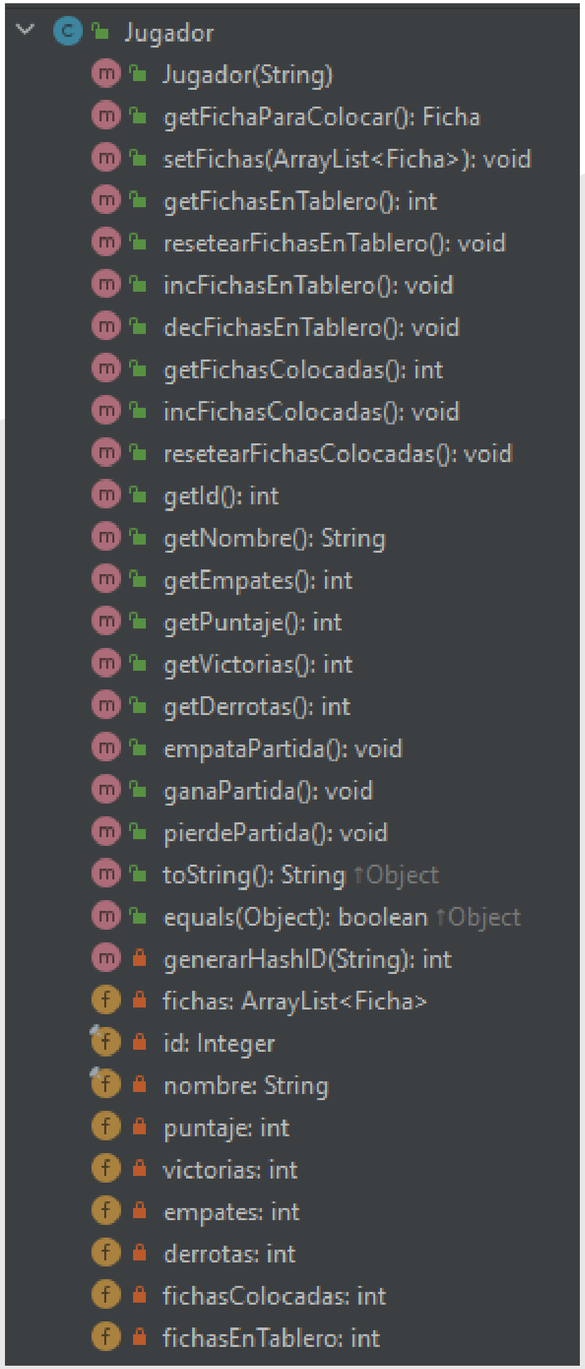
Cada ficha conoce a qué jugador corresponde.

Clases


Tablero
Tablero()
getCasilla(Coordenada): Casilla
obtenerEstadoCasilla(Coordenada): EstadoCasilla
colocarFicha(Coordenada, Ficha): void
quitarFicha(Coordenada): void
moverFicha(Coordenada, Coordenada): void
obtenerFicha(Coordenada): Ficha
obtenerCasillasOcupadasPorJugador(Jugador): List<Casilla>
inicializarCasillas(): void
conectarCasillas(): void
casillas: ArrayList<Casilla>


Coordenada
Coordenada(int, int)
getFila(): int
getColumna(): int
equals(Object): boolean ↑Object
fila: int
columna: int


Ficha
Ficha(Jugador)
getJugador(): Jugador
jugador: Jugador


Jugador
Jugador(String)
getFichaParaColocar(): Ficha
setFichas(ArrayList<Ficha>): void
getFichasEnTablero(): int
resetearFichasEnTablero(): void
incFichasEnTablero(): void
decFichasEnTablero(): void
getFichasColocadas(): int
incFichasColocadas(): void
resetearFichasColocadas(): void
getId(): int
getNombre(): String
getEmpates(): int
getPuntaje(): int
getVictorias(): int
getDerrotas(): int
empataPartida(): void
ganaPartida(): void
pierdePartida(): void
toString(): String ↑Object
equals(Object): boolean ↑Object
generarHashID(String): int
fichas: ArrayList<Ficha>
id: Integer
nombre: String
puntaje: int
victorias: int
empates: int
derrotas: int
fichasColocadas: int
fichasEnTablero: int

Patrones de diseño

Veamos cómo se implementaron para desarrolar este juego

Patrón MVC

¿Quiénes son las clases involucradas?

Patrón Observer

¿Quién notifica a quién? ¿Cómo y cuándo?

Patrón MVC

MODELO: Molino

VISTA: IVista

CONTROLADOR: Controlador

El modelo es Molino. En esta clase es donde se gestiona toda la lógica del juego. Desde la gestión de los turnos hasta la verificación de si se produjo o no un molino. Molino conoce al Tablero, y es allí donde se interactúa con las fichas y los jugadores. Molino conoce a una clase “ReglasDelJuego” en donde se recorre al tablero para analizar todos los casos que se puedan producir dentro del tablero.

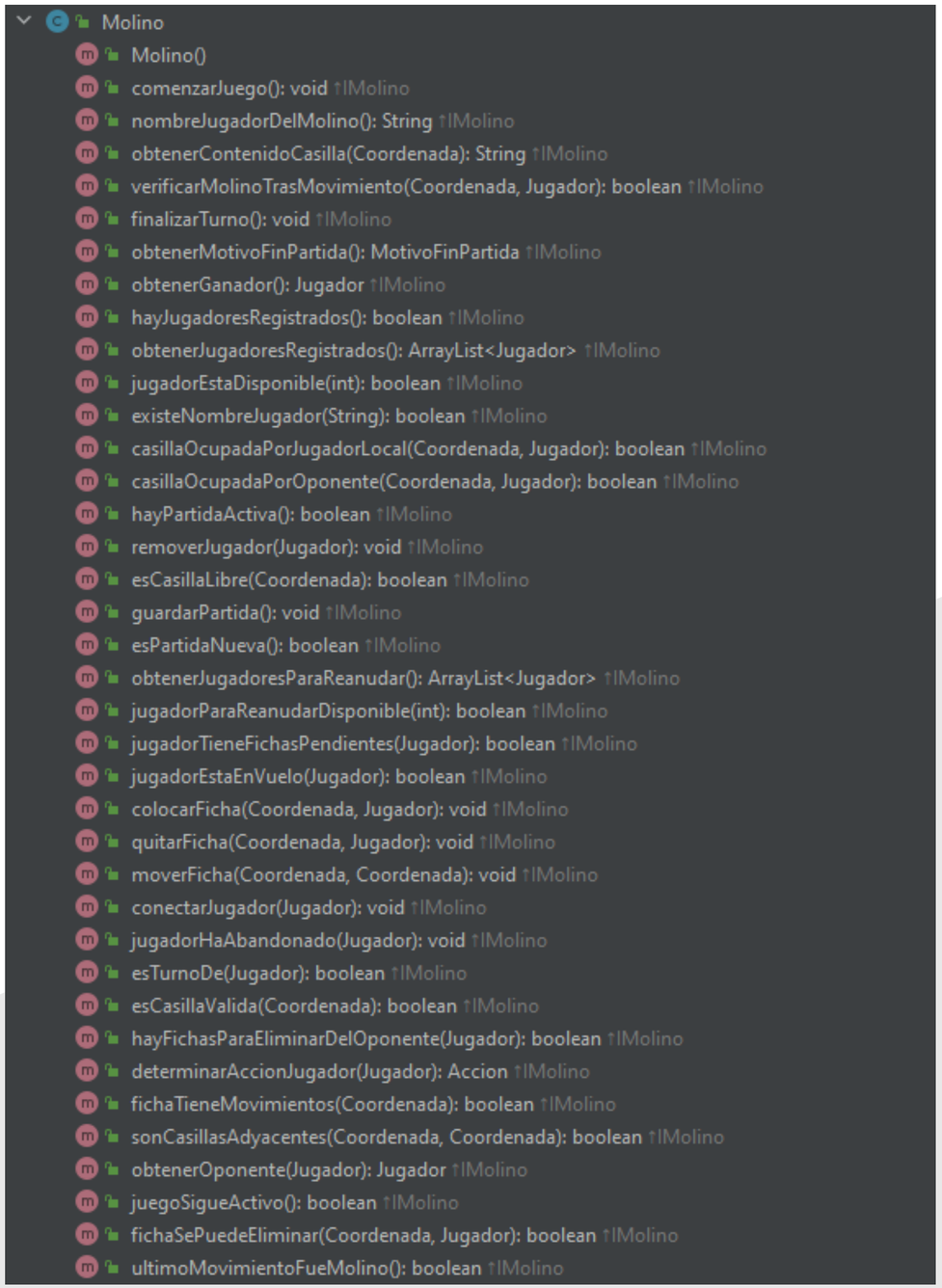
En la vista, nos vamos a encontrar con 3 interfaces visuales:

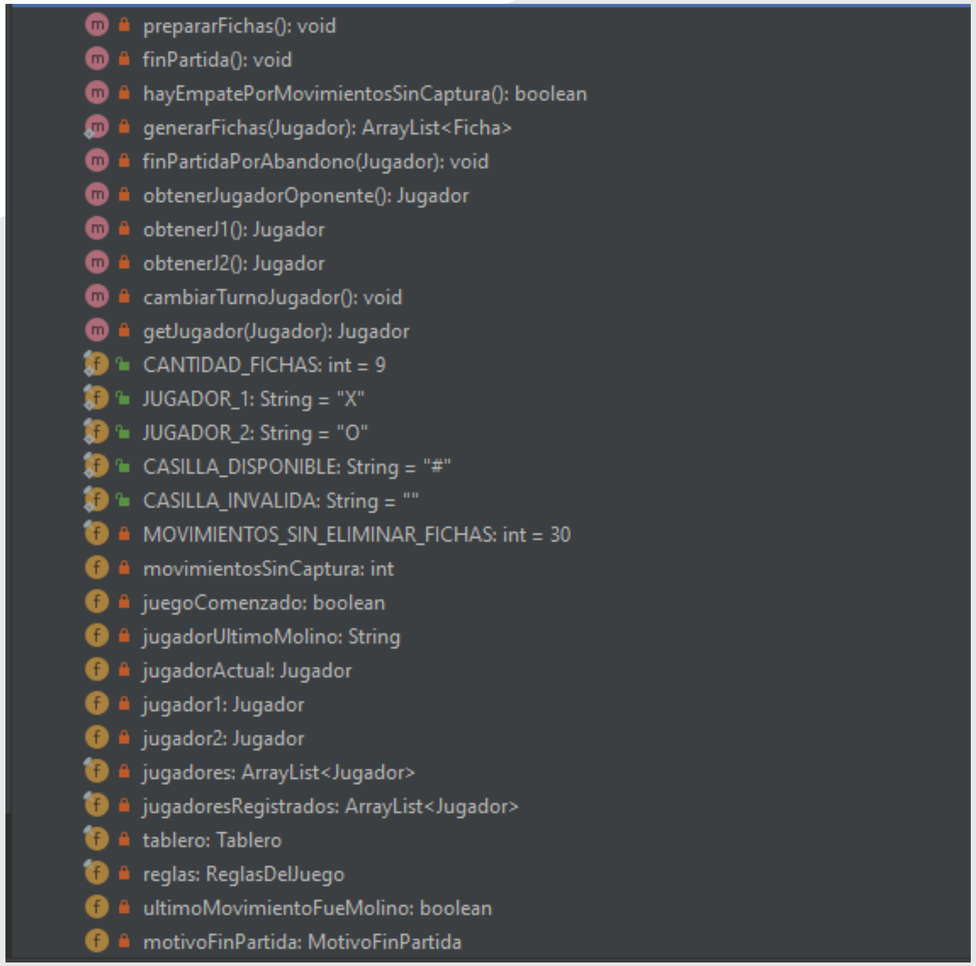
- Vista de consola
- Vista de consola algo mejorada
- Vista de interfaz gráfica con imágenes

Todas las vistas creadas implementan y funcionan con la misma interfas “IVista”.

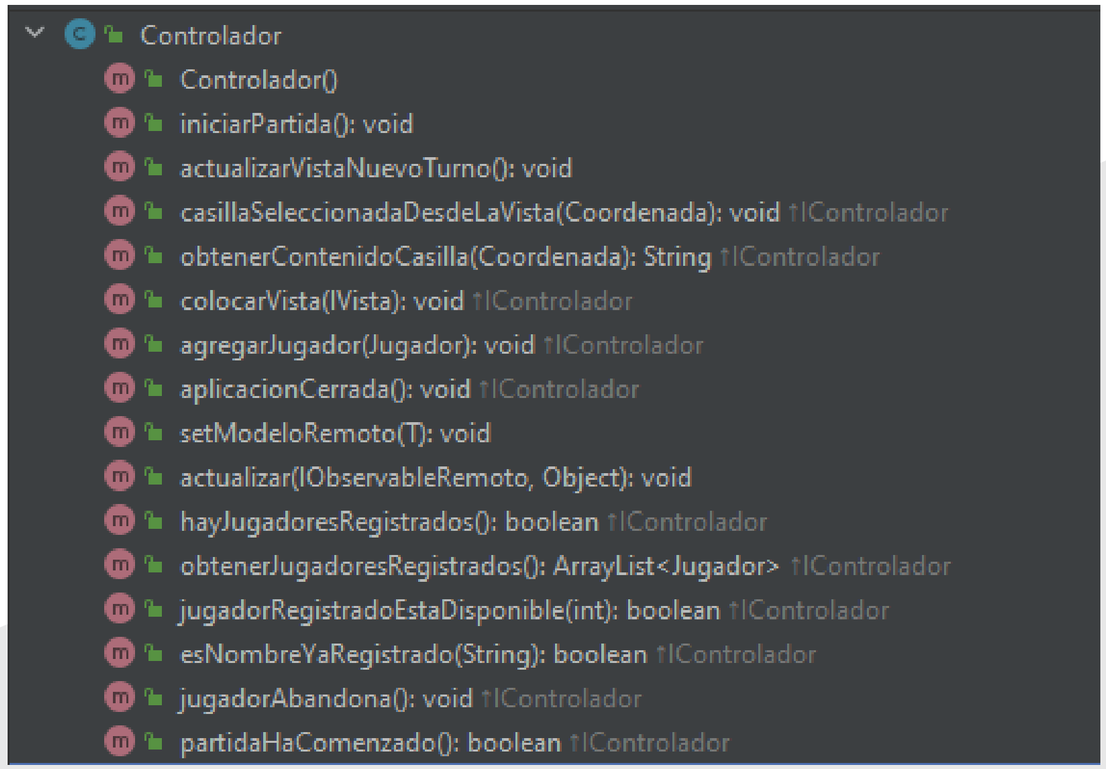
El controlador es quién le realiza todas las consultas a Molino, para saber si es el turno del jugador. También se le consulta si hubo o no molino, para saber cómo seguir con el flujo del juego.
Conoce a “IVista” y se encarga de enviar todos los cambios necesarios en base a la situación del juego.

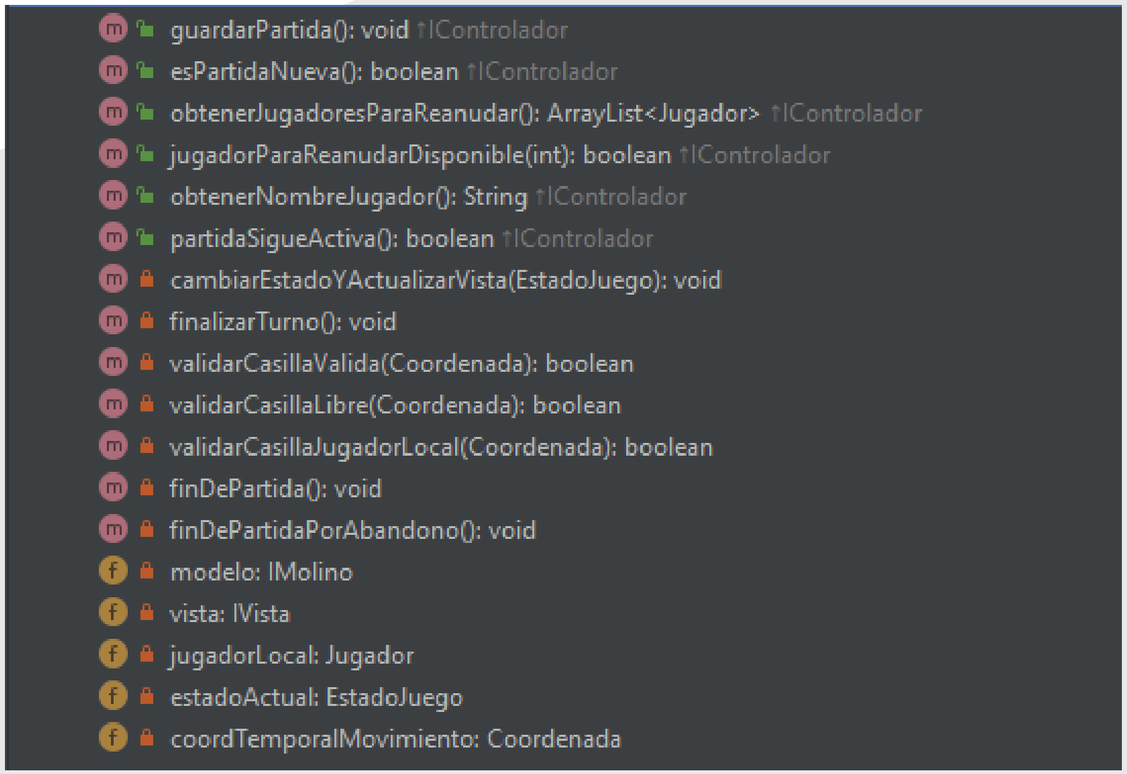
Molino

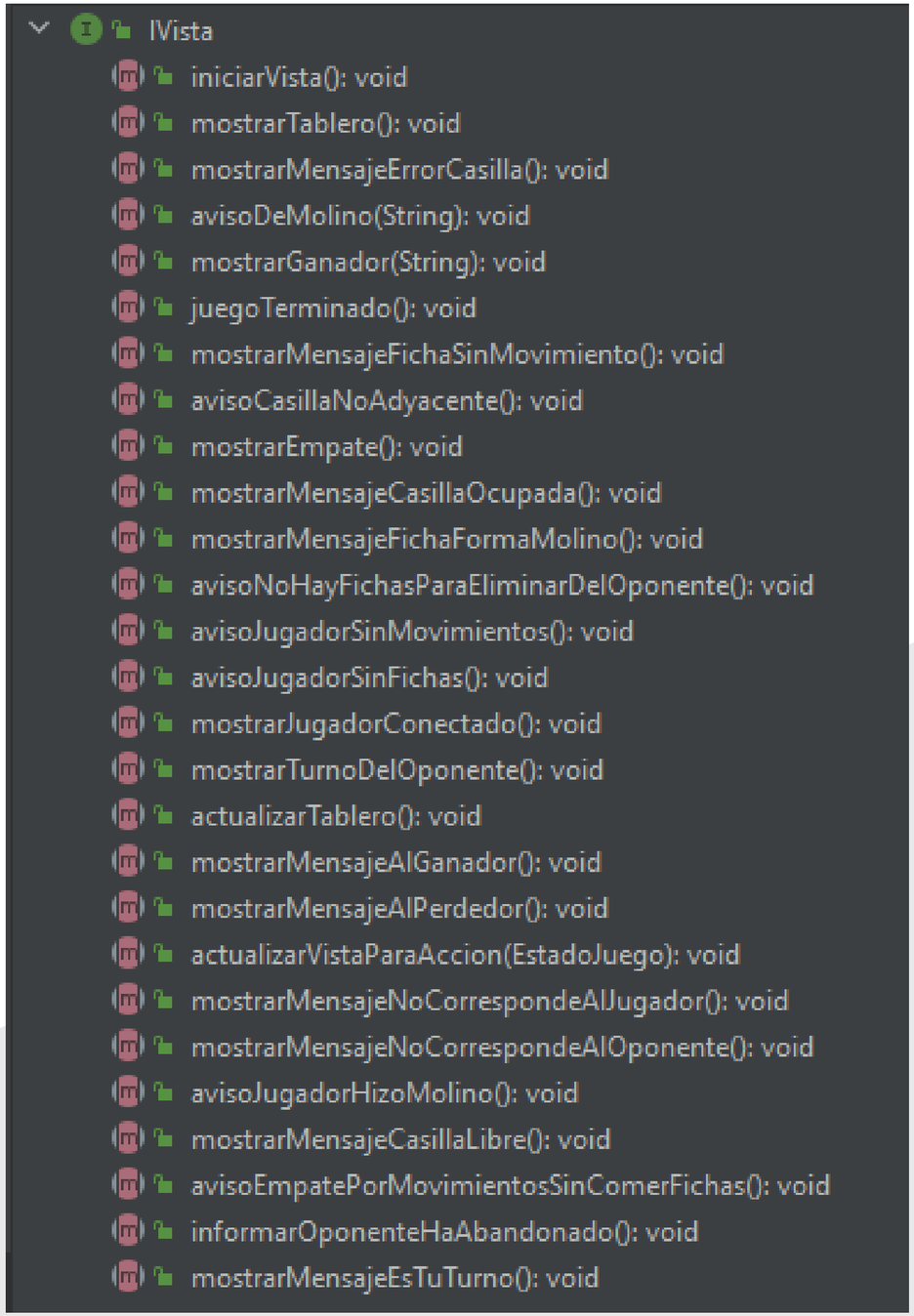

Molino
Molino()
comenzarJuego(): void ↑IMolino
nombreJugadorDelMolino(): String ↑IMolino
obtenerContenidoCasilla(Coordenada): String ↑IMolino
verificarMolinoTrasMovimiento(Coordenada, Jugador): boolean ↑IMolino
finalizarTurno(): void ↑IMolino
obtenerMotivoFinPartida(): MotivoFinPartida ↑IMolino
obtenerGanador(): Jugador ↑IMolino
hayJugadoresRegistrados(): boolean ↑IMolino
obtenerJugadoresRegistrados(): ArrayList<Jugador> ↑IMolino
jugadorEstaDisponible(int): boolean ↑IMolino
existeNombreJugador(String): boolean ↑IMolino
casillaOcupadaPorJugadorLocal(Coordenada, Jugador): boolean ↑IMolino
casillaOcupadaPorOponente(Coordenada, Jugador): boolean ↑IMolino
hayPartidaActiva(): boolean ↑IMolino
removerJugador(Jugador): void ↑IMolino
esCasillaLibre(Coordenada): boolean ↑IMolino
guardarPartida(): void ↑IMolino
esPartidaNueva(): boolean ↑IMolino
obtenerJugadoresParaReanudar(): ArrayList<Jugador> ↑IMolino
jugadorParaReanudarDisponible(int): boolean ↑IMolino
jugadorTieneFichasPendientes(Jugador): boolean ↑IMolino
jugadorEstaEnVuelo(Jugador): boolean ↑IMolino
colocarFicha(Coordenada, Jugador): void ↑IMolino
quitarFicha(Coordenada, Jugador): void ↑IMolino
moverFicha(Coordenada, Coordenada): void ↑IMolino
conectarJugador(Jugador): void ↑IMolino
jugadorHaAbandonado(Jugador): void ↑IMolino
esTurnoDe(Jugador): boolean ↑IMolino
esCasillaValida(Coordenada): boolean ↑IMolino
hayFichasParaEliminarDelOponente(Jugador): boolean ↑IMolino
determinarAccionJugador(Jugador): Accion ↑IMolino
fichaTieneMovimientos(Coordenada): boolean ↑IMolino
sonCasillasAdyacentes(Coordenada, Coordenada): boolean ↑IMolino
obtenerOponente(Jugador): Jugador ↑IMolino
juegoSigueActivo(): boolean ↑IMolino
fichaSePuedeEliminar(Coordenada, Jugador): boolean ↑IMolino
ultimoMovimientoFueMolino(): boolean ↑IMolino


prepararFichas(): void
finPartida(): void
hayEmpatePorMovimientosSinCaptura(): boolean
generarFichas(Jugador): ArrayList<Ficha>
finPartidaPorAbandono(Jugador): void
obtenerJugadorOponente(): Jugador
obtenerJ1(): Jugador
obtenerJ2(): Jugador
cambiarTurnoJugador(): void
getJugador(Jugador): Jugador
CANTIDAD_FICHAS: int = 9
JUGADOR_1: String = "X"
JUGADOR_2: String = "O"
CASILLA_DISPONIBLE: String = "#"
CASILLA_INVALIDA: String = ""
MOVIMIENTOS_SIN_ELIMINAR_FICHAS: int = 30
movimientosSinCaptura: int
juegoComenzado: boolean
jugadorUltimoMolino: String
jugadorActual: Jugador
jugador1: Jugador
jugador2: Jugador
jugadores: ArrayList<Jugador>
jugadoresRegistrados: ArrayList<Jugador>
tablero: Tablero
reglas: ReglasDelJuego
ultimoMovimientoFueMolino: boolean
motivoFinPartida: MotivoFinPartida

Controlador






IVista
iniciarVista(): void
mostrarTablero(): void
mostrarMensajeErrorCasilla(): void
avisoDeMolino(String): void
mostrarGanador(String): void
juegoTerminado(): void
mostrarMensajeFichaSinMovimiento(): void
avisoCasillaNoAdyacente(): void
mostrarEmpate(): void
mostrarMensajeCasillaOcupada(): void
mostrarMensajeFichaFormaMolino(): void
avisoNoHayFichasParaEliminarDelOponente(): void
avisoJugadorSinMovimientos(): void
avisoJugadorSinFichas(): void
mostrarJugadorConectado(): void
mostrarTurnoDelOponente(): void
actualizarTablero(): void
mostrarMensajeAlGanador(): void
mostrarMensajeAlPerdedor(): void
actualizarVistaParaAccion(EstadoJuego): void
mostrarMensajeNoCorrespondeAlJugador(): void
mostrarMensajeNoCorrespondeAlOponente(): void
avisoJugadorHizoMolino(): void
mostrarMensajeCasillaLibre(): void
avisoEmpatePorMovimientosSinComerFichas(): void
informarOponenteHaAbandonado(): void
mostrarMensajeEsTuTurno(): void

IVista

Patrón Observer

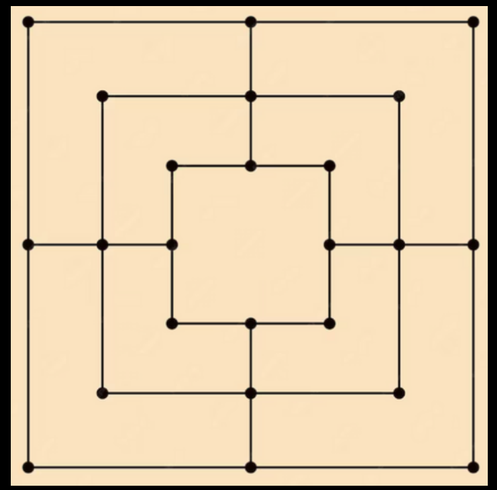
Para llevarlo a cabo, se utilizaron las herramientas brindadas con la librería de RMI.

¿Quién es el observable? La clase Molino.
Cuando se produzca algún cambio en el tablero, ya sea que se hizo un movimiento, se colocó o quitó una ficha, Molino es quien se encarga de notificar a sus Observadores sobre el cambio que se produjo en el modelo.

¿Quién es el observador? El controlador.
El controlador es quién está pendiente a los cambios del modelo.
Cuando recibe el cambio, se encarga de mandarle la orden a la vista para que muestre una nueva información por pantalla. Sea un mensaje o actualizar el estado del tablero.

Interfaces visuales implementadas

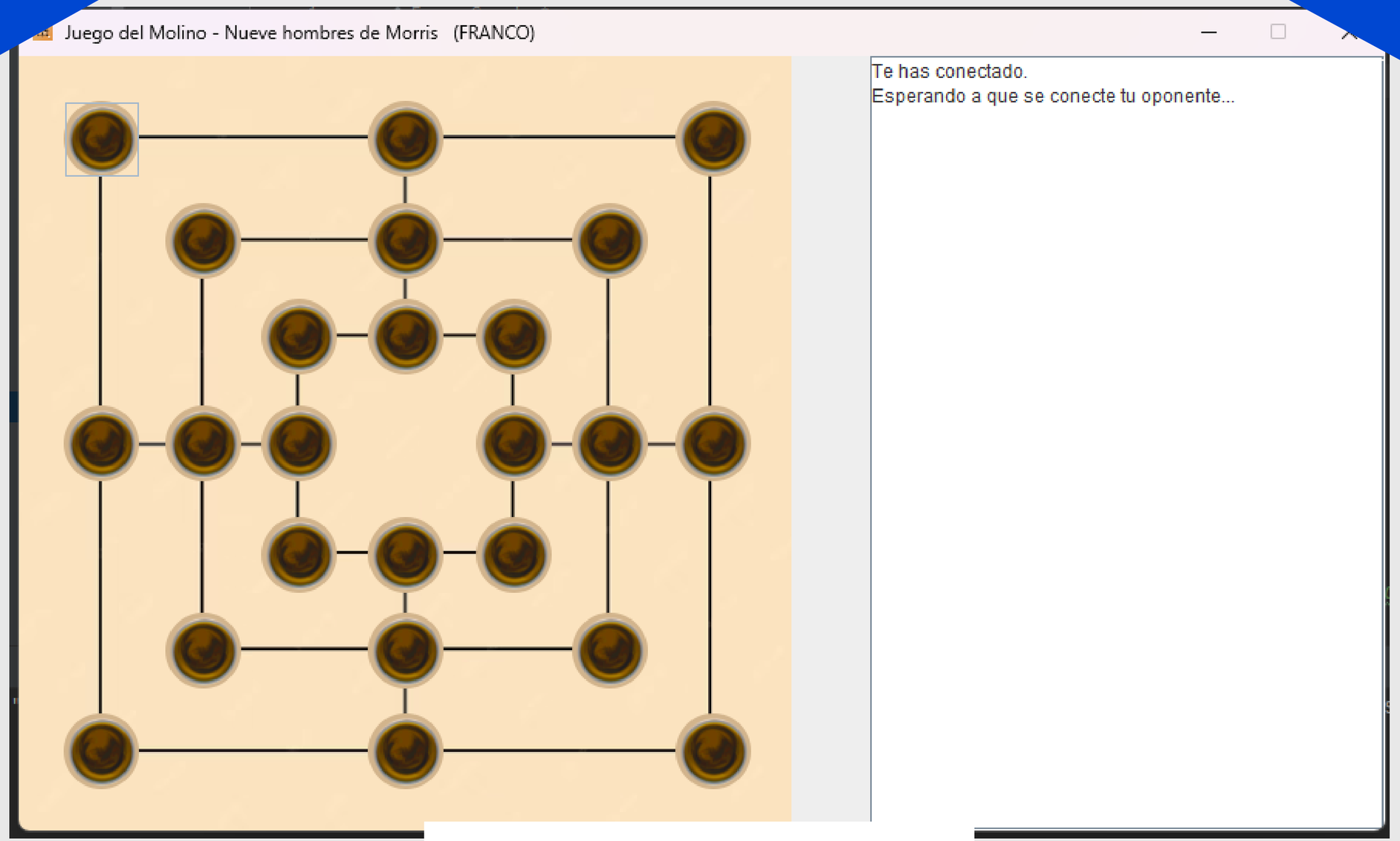
Veremos cuáles son las interfaces implementadas y cómo funcionan cada una



Interfaz gráfica

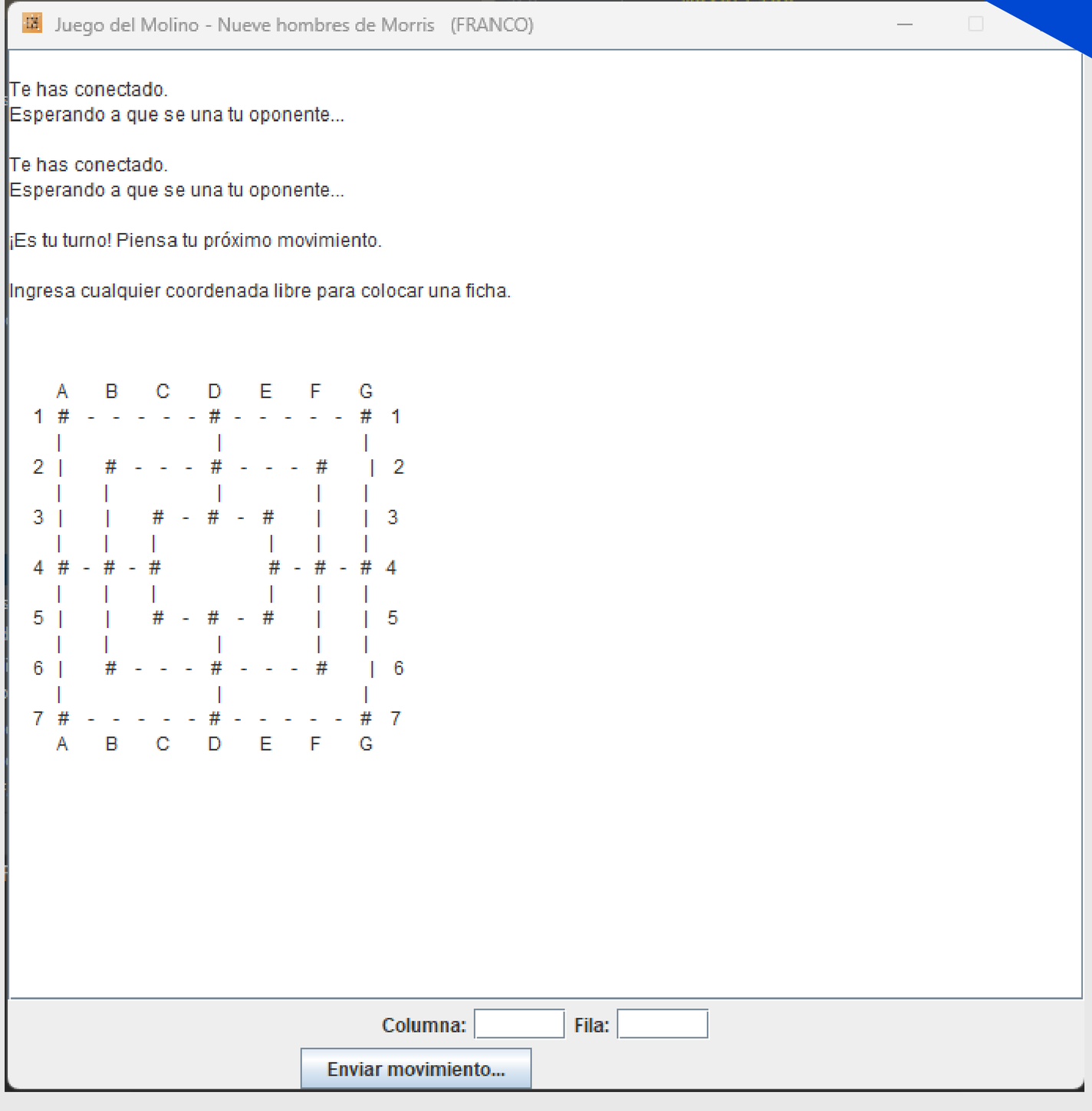
Interfaz de consola

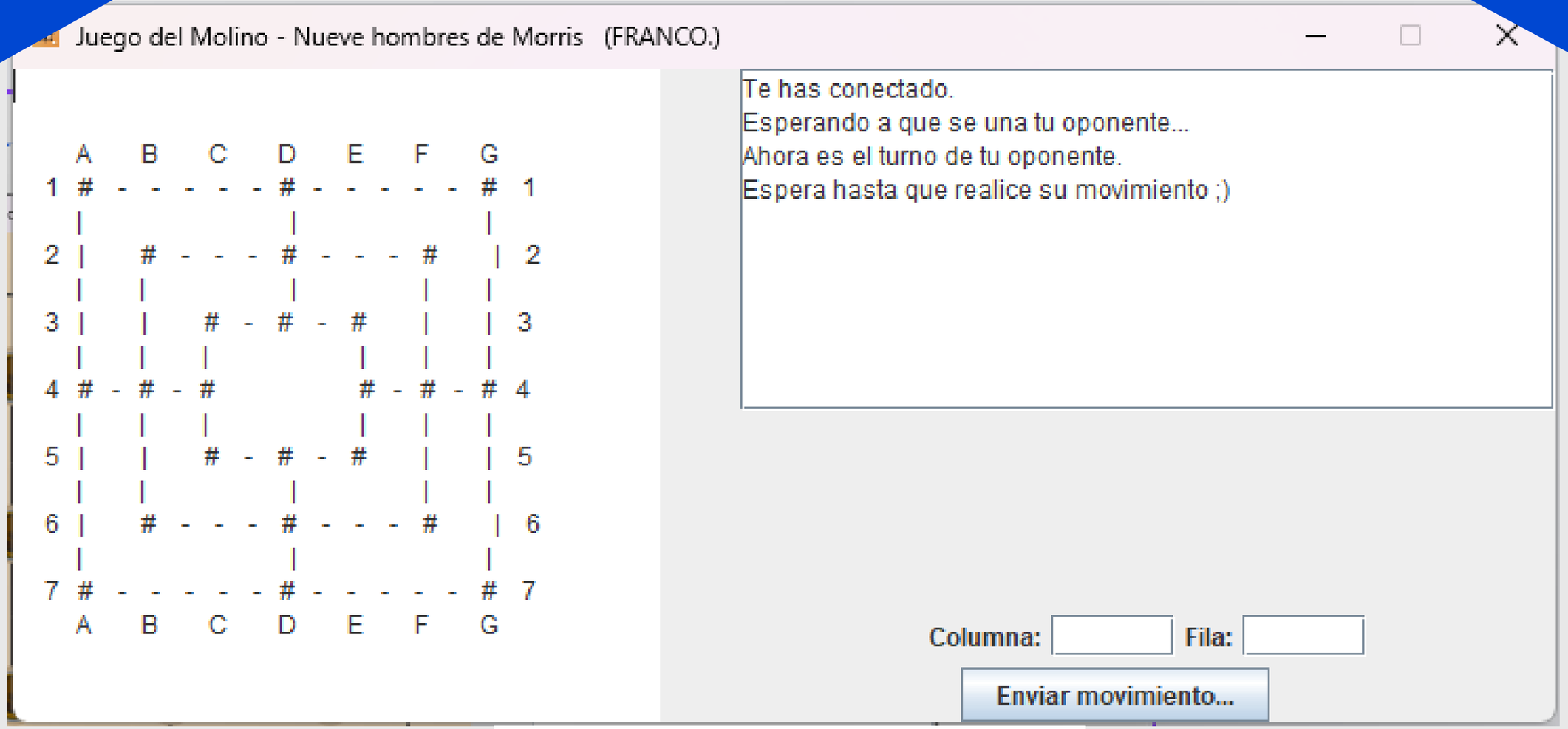
Interfaz de consola (mejorada)



INTERFAZ GRÁFICA

INTERFAZ DE CONSOLA





INTERFAZ DE CONSOLA (Mejorada)

Funcionamiento en red y persistencia de información

¿Cómo funcionan estos dos puntos en el juego?

Juego en red

´Para crear o reanudar una partida, se tiene que iniciar un servidor. Una vez creado, desde otra computadora u otra aplicacion corriendo se tiene que ejecutar el cliente, en donde se estará utilizando la IP de la computadora que ejecuta el servidor.

Persistencia

- Cuando un jugador es creado, este ya se guarda. Luego se va actualizando a medida que vaya jugando partida, si es que es seleccionado.
- Cuando un jugador quiera abandonar una partida que está en juego, podrá directamente abandonar o guardar el estado actual de la partida. Al guardarla, luego se podrá elegir para reanudarla.

¡Muchas gracias!

Presentado y desarrollado por:
Franco M. Saracho

Datos de interés:

Email
fmsaracho64@gmail.com

Enlace al repositorio
https://github.com/FrancoSaracho64/Molino