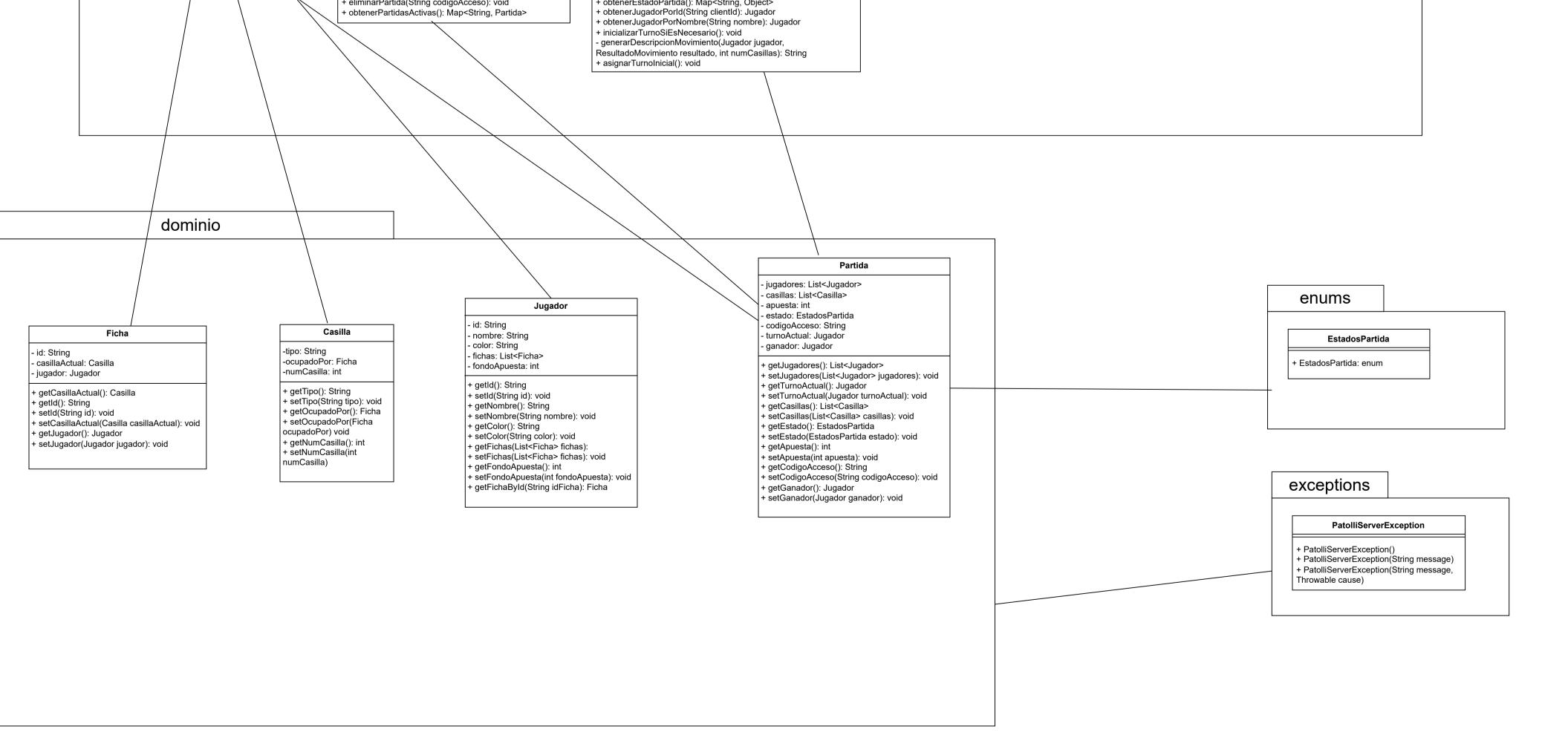
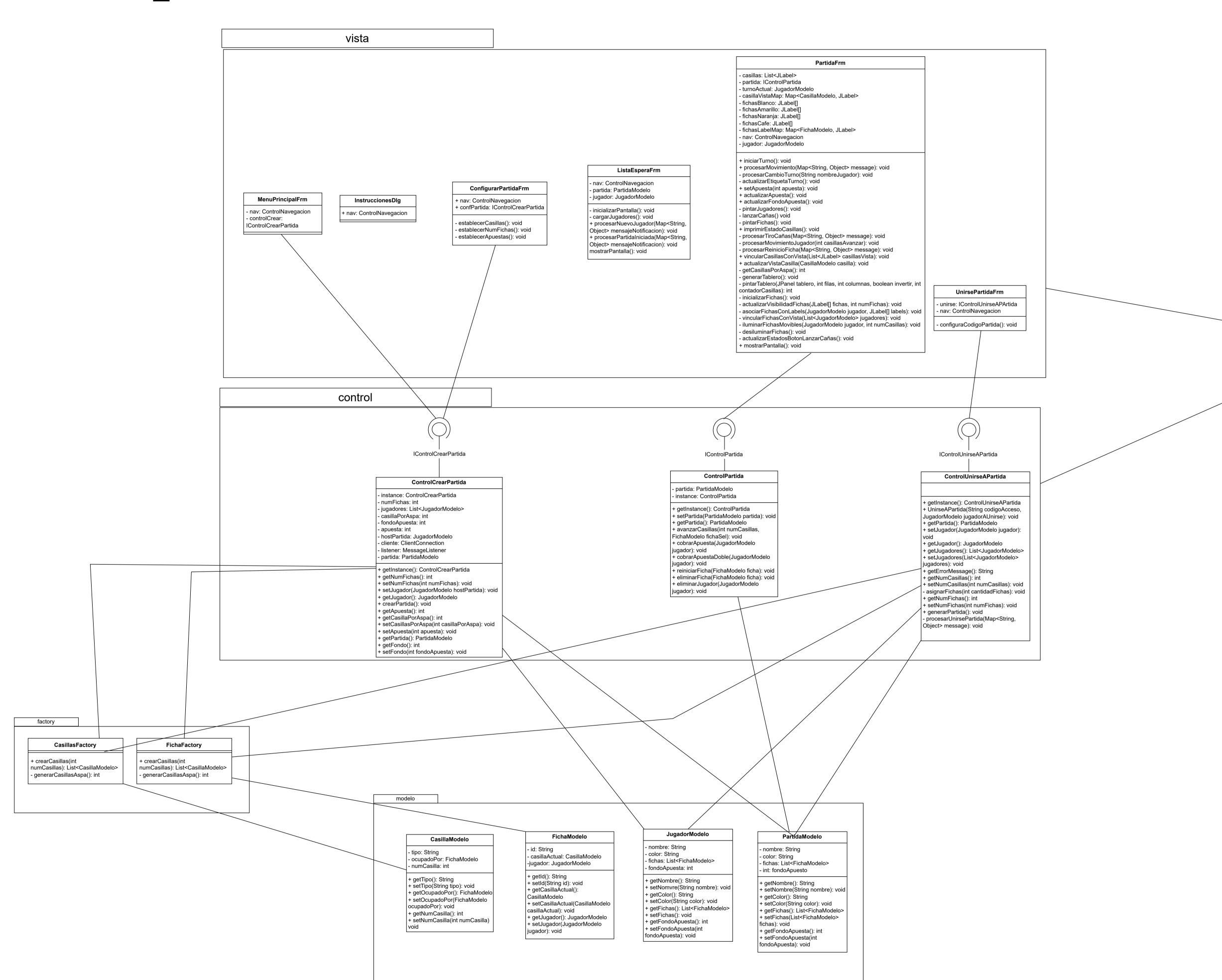
Patolli_Commons Patolli_Servidor comunicacion ClientHandler - clientSocket: Socket - handlerActions: HandlerActions Servidor - PUERTO: int - threadPool: ExecutorServide - handlerActions: HandlerActions - clientCounter: int - jugadores: List<Jugador> - turnoActual: Jugador - clientSockets: List<Socket> HandlerActions - gestionarPartidaBO: GestionarPartidaBO - clientId: String MessageUtil + handle(Socket clientSocket, Map<String, Object> + handle(Socket clientSocket, Map<String, Object> data): void - handleCrearPartida(Socket clientSocket, Map<String, Object> data): void - handleUnirsePartida(Socket clientSocket, Map<String, Object> data): void - handleIniciarPartida(Socket clientSocket, Map<String, Object> data): void - handleTirarCaña(Socket clientSocket, Map<String, Object> data): void - handleMoverFicha(Socket clientSocket, Map<String, Object> data): void - handleReiniciarFicha(Socket clientSocket, Map<String, Object> data): void - handleReiniciarFicha(Socket clientSocket, Map<String, Object> data): void - enviarError(Socket clientSocket, String mensajeError): void + run(): void - cerrarConexion(): void + enviarMensaje(Socket clientSocket, Map<String, Object> mensajeMap): void + iniciar(): void + detener(): void + main ClientManager - clientToldMap: ConcurrentHashMap<>() - idToClientMap: ConcurrentHashMap<>() - clientIdToJugadorMap: ConcurrentHashMap<> + addClient(Socket clientSocket, String clientId, Jugador jugador): void + getJugadorByClientId(String clientId): Jugador + getClientSocket(String cliendId): Socket + removeClient(Socket clientSocket): void + getOtherPlayer(String excludeClientId): Jugador - handleCambiarTurno(Socket clientSocket, Map<String, Object> data): void - obtenerClientId(Socket clientSocket): String CrearPartidaBO PartidaLogicaBO GestionarPartidaBO - instance: GestionarPartidaBO - partidasActivas: Map<String, Partida> + moverFicha(Map<String, Object> data, String clientId): Map<String, + crearPartida(Map<String, Object> data, String clientId): Map<String, Object> - crearCasillas(int casillasTotales): List<Casilla> - generarCasillasAspa(List<Casilla> casillas, int + getInstance(): GestionarPartidaBO + registrarPartida(Partida partida): void + unirseAPartida(Map<String, Object> data, String clientId): Map<String, Object> ResultadoMovimientoDetallado + lanzamientoCañas(String clientId): Map<String, Object> + reiniciarFichaPorCliente(Map<String, Object> data, String clientId): Map<String, Object> + cambiarTurno(String clientId): Map<String, Object> + avanzaCasillas(int numCasillas, Ficha fichaSel): - resultado: ResultadoMovimiento - jugadorEliminado: Jugador contadorCasilla, int casillasPorAspa, String inicialJugador, int apuestaPos1, int apuestaPos2, int dobleTurnoPos1, int dobleTurnoPos2): int - crearNuevoJugador(String clientId, String nombre, Partida partida, Jugador jugadorBase): Jugador - notificarJugadores (Partida partida, Jugador + gerResultado(): ResultadoMovimiento + getJugadorEliminado(): Jugador - crearJugador(String clientId, String nombre, int fondo, ResultadoMovimiento nuevoJugador): void - crearRespuestaUnirsePartida(Partida partida, Jugador nuevoJugador, Jugador jugadorBase): Map<String, Object> + iniciarPartida(Map<String, Object> data, String clientId): Map<String, Object> - validarCliente(String clientId): void - esCasillaGanadora(Ficha ficha, Casilla casillaDestino): boolear + cobrarApuesta(Jugador jugador): boolean + reiniciarFicha(Ficha ficha): void - crearFichas(int numFichas, Jugador jugador): List<Ficha> + getPartida(): Partida + setPartida(Partida partida): void + eliminarFicha(Ficha ficha): void + eliminarJugador(Jugador jugador): void + determinarTurno(): Jugador + partidaFinalizadas(): boolean - generarCodigoNumerico(int longitud): String - asignarColor(Partida partida): String - crearFichas(int numFichas, Jugador jugador): List<Ficha> + obtenerPartida(String codigoAcceso): Partida - validarCliente(String clientId): void + tirarCañas(): Map<String, Object> - validarCliente(String clientId): void + obtenerEstadoPartida(): Map<String, Object> + obtenerJugadorPorld(String clientId): Jugador + obtenerPartidasActivas(): Map<String, Partida> + obtenerJugadorPorNombre(String nombre): Jugador + inicializarTurnoSiEsNecesario(): void - generarDescripcionMovimiento(Jugador jugador, ResultadoMovimiento resultado, int numCasillas): String + asignarTurnoInicial(): void dominio - jugadores: List<Jugador> enums - casillas: List<Casilla>



Patolli_Cliente



ClientConnection

+ unirseAPartida(String codigoAcceso, JugadorModelo jugadorAUnirse): void + iniciarPartida(String codigoAcceso): void + lanzarCaña(String codigoPartida): void

+ reiniciarFicha(String codigoAcceso, String fichald): void + cambiarTurno(String codigoPartida): void + moverFicha(String codigoPartida,String idFicha,int numCasillas): void + setMessageListener(MessageListener listener): void + MessageListener

- instance: ClientConnection

- outputStream: OutputStream
- inputStream: InputStream
- listeningThread: Thread
- running: boolean
- objectMapper: ObjectMapper
- messageListener: MessageListener

+ disconnect(): void

startListening(): void

+ getInstance(): ClientConnection + connect(String host, int port): boolean

+ sendMessage(Map<String, Object> data): void + crearPartida(PartidaModelo partida): void