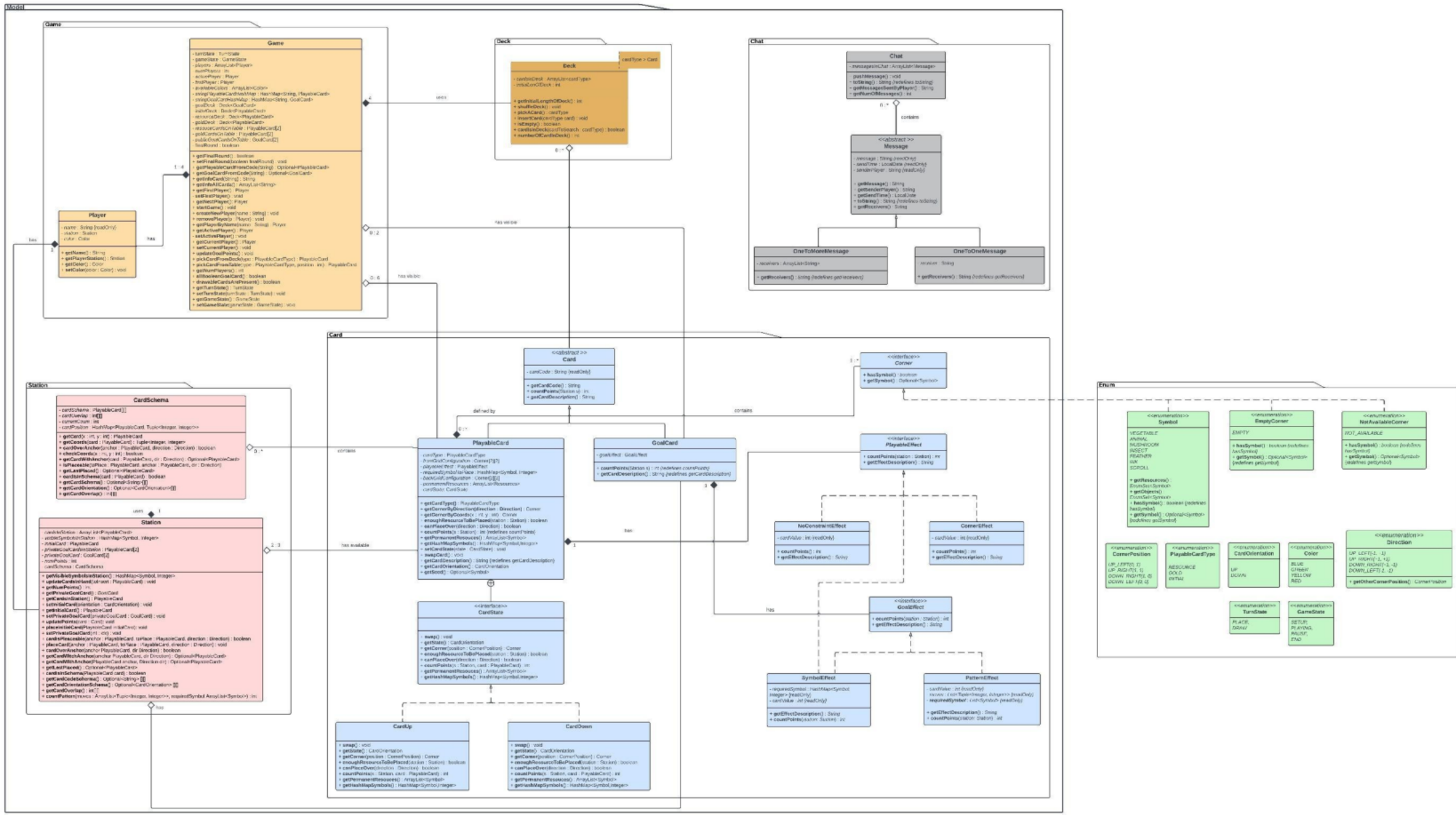
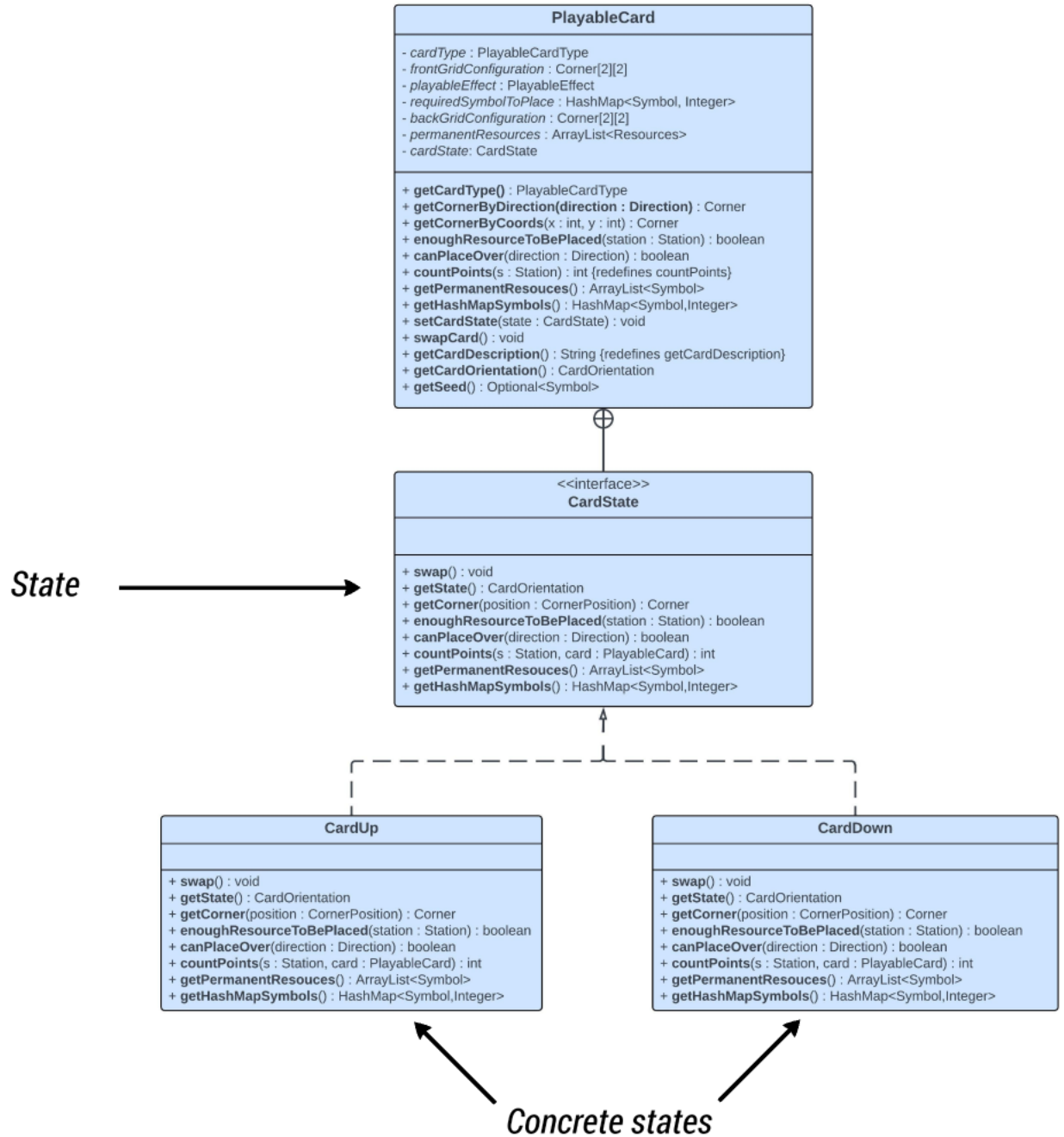
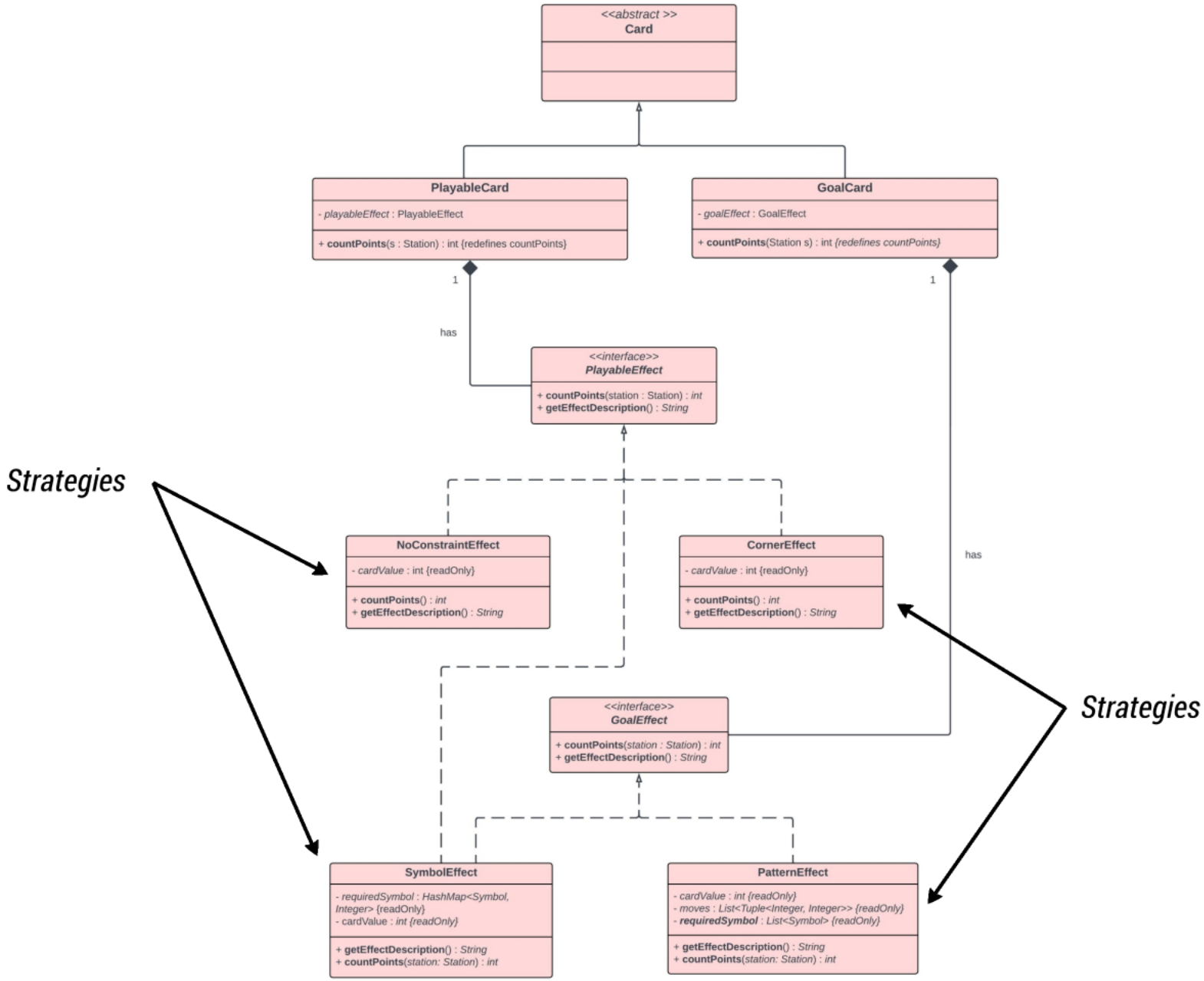
Our Model...


Model
Game
Game
Player
Deck
Deck
Chat
Chat
Message
OneToMoreMessage
OneToOneMessage
Card
Card
PlayableCard
GoalCard
Corner
CardState
CardUp
CardDown
Station
CardSchema
Station
Enum
Symbol
EmptyCorner
NotAvailableCorner
CornerPosition
PlayableCardType
CardOrientation
Color
Direction
TurnState
GameState
PlayableEffect
NoConstraintEffect
CornerEffect
GoalEffect
SymbolEffect
PatternEffect

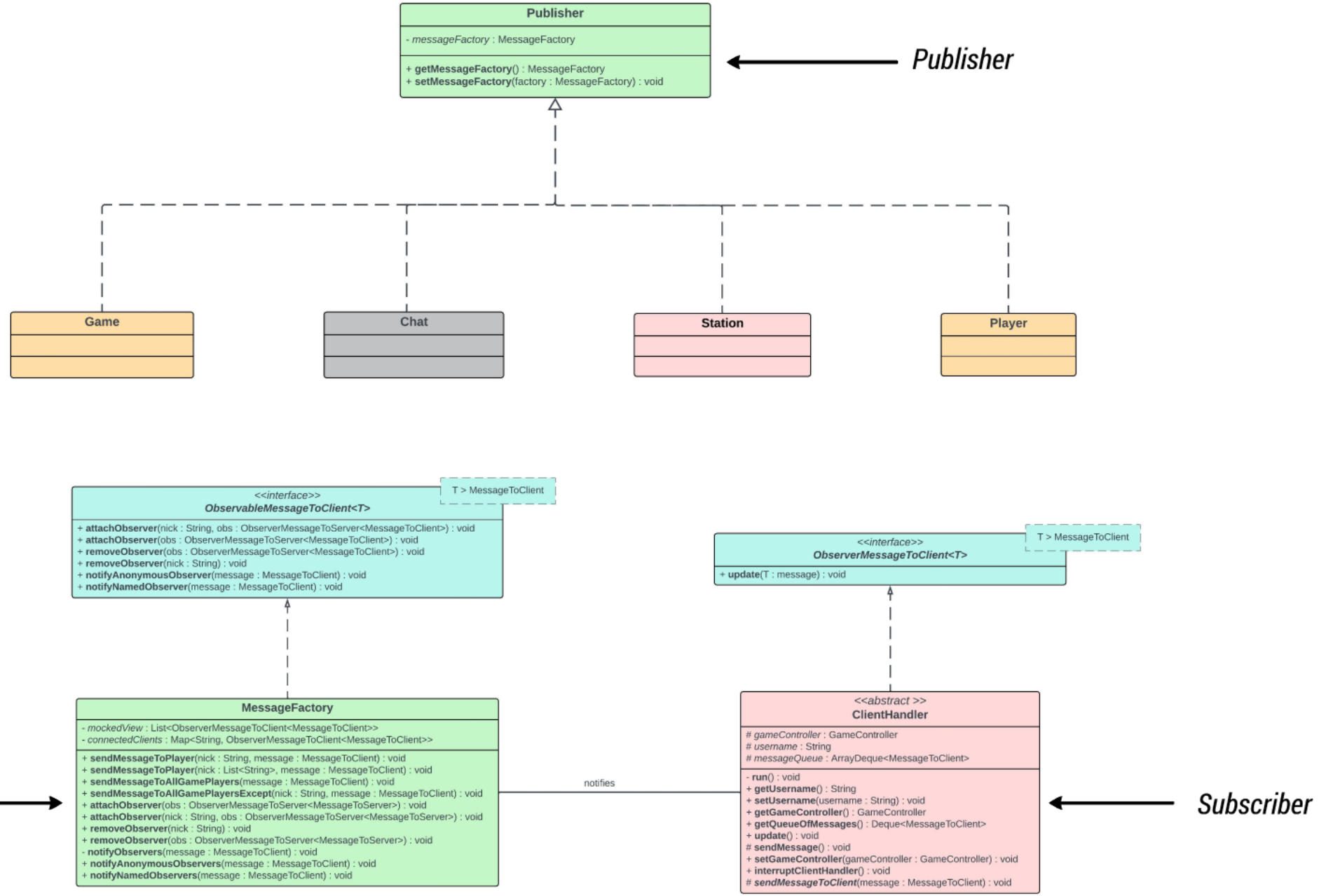
State Pattern



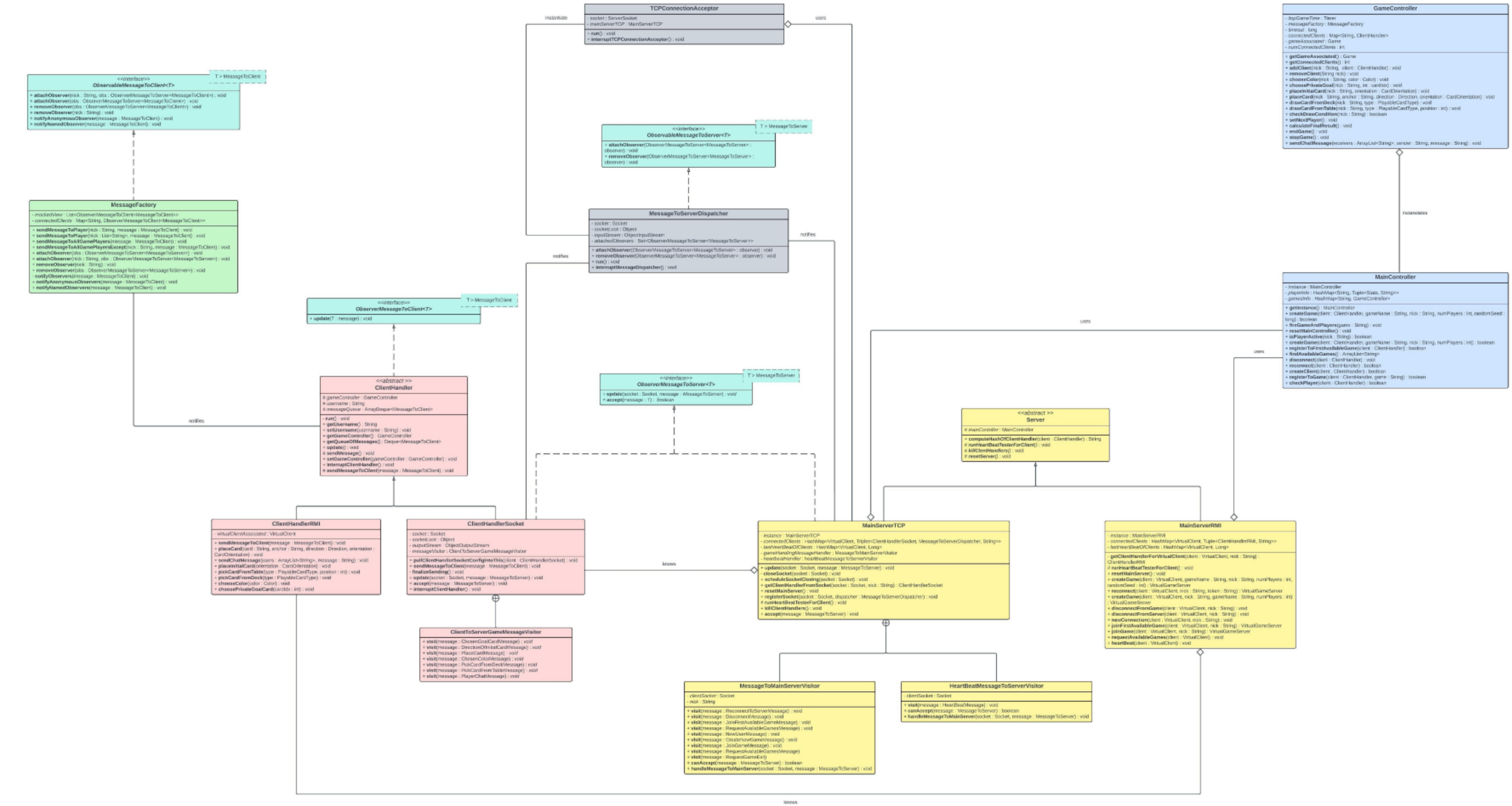
Strategy Pattern



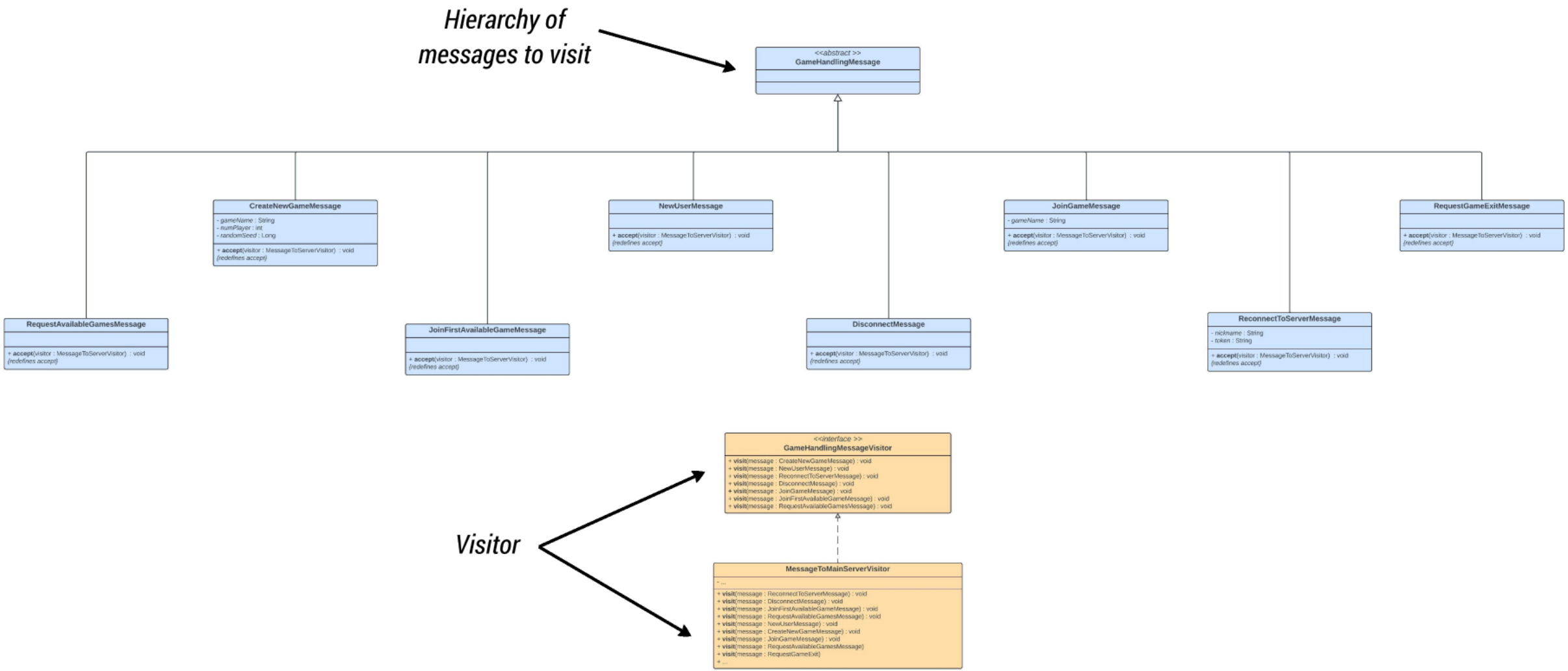
Publisher - Subscriber Pattern



Our Server and Controller...

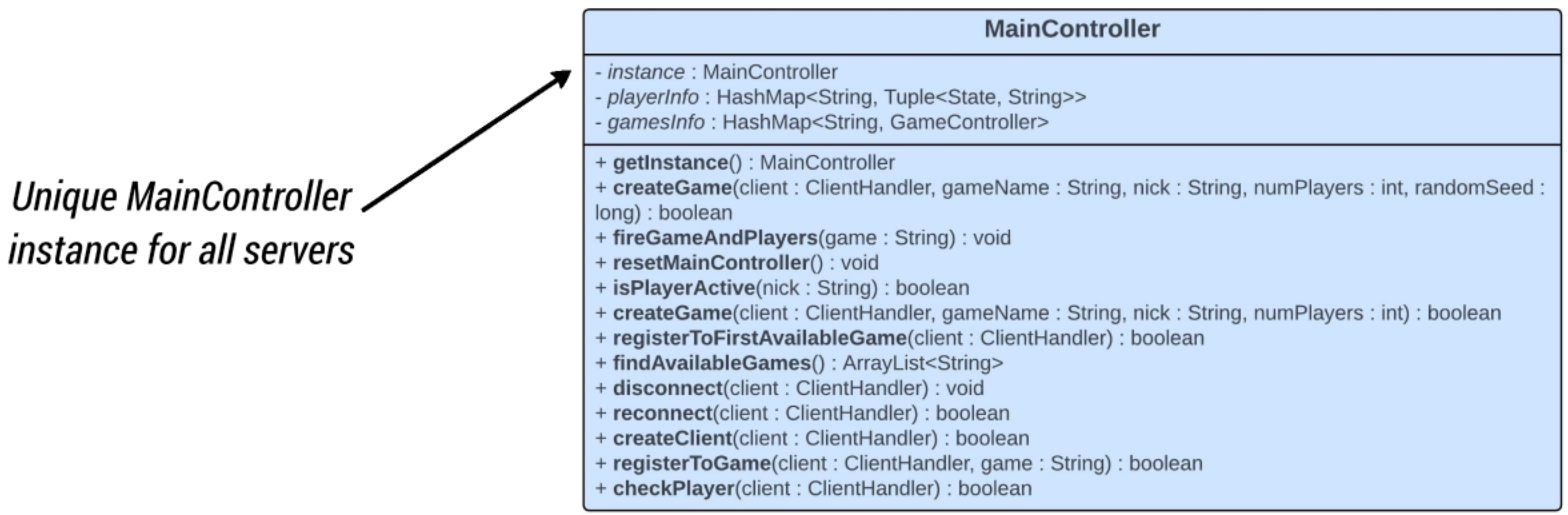


Visitor Pattern

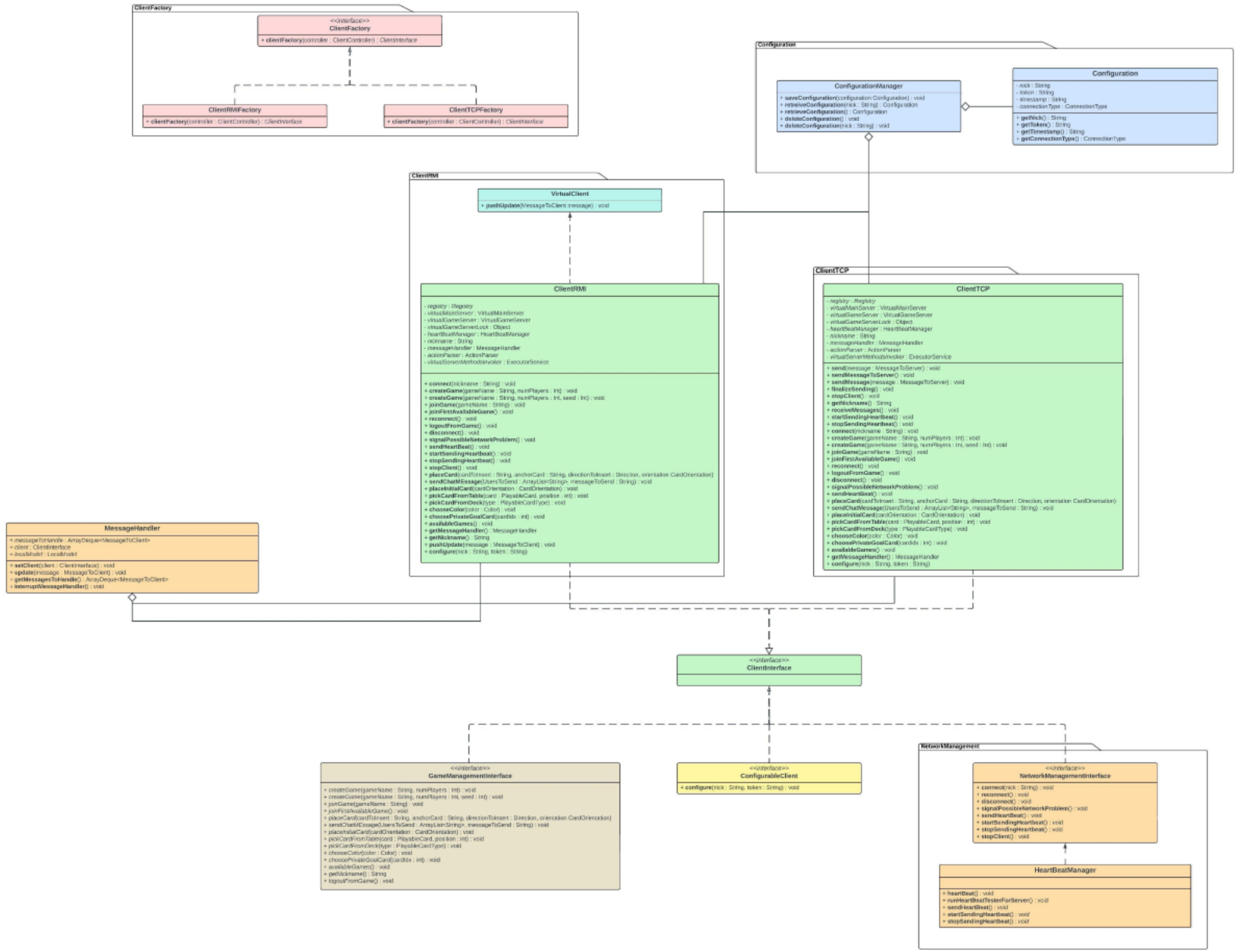


Singleton Pattern

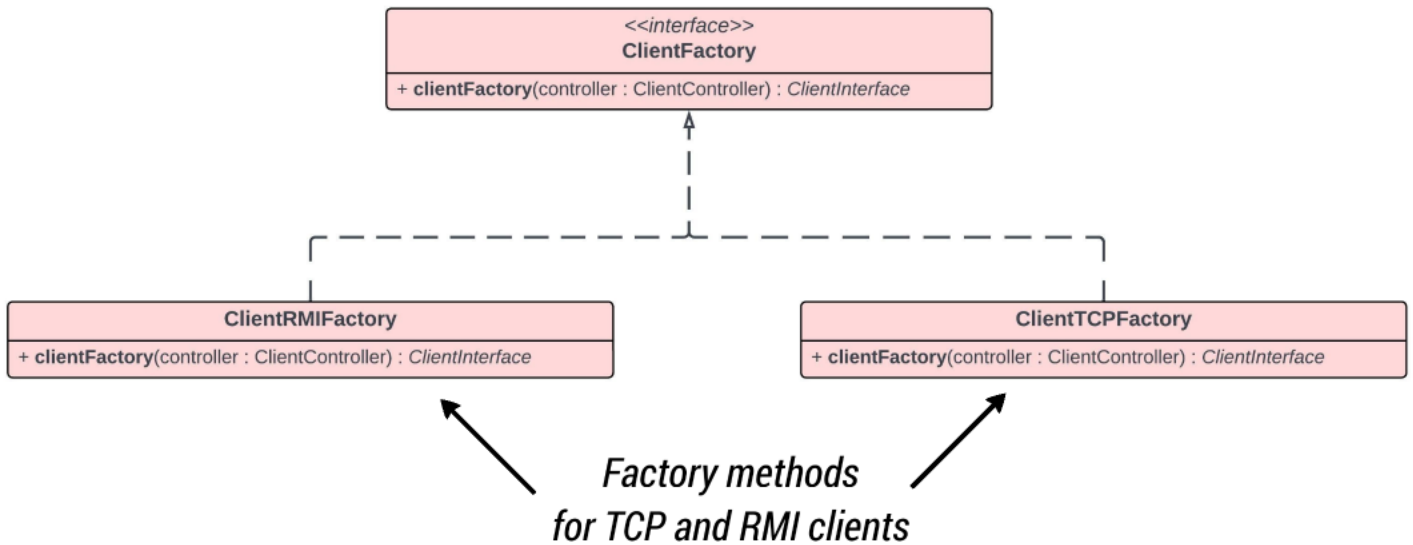
Unique MainController instance for all servers



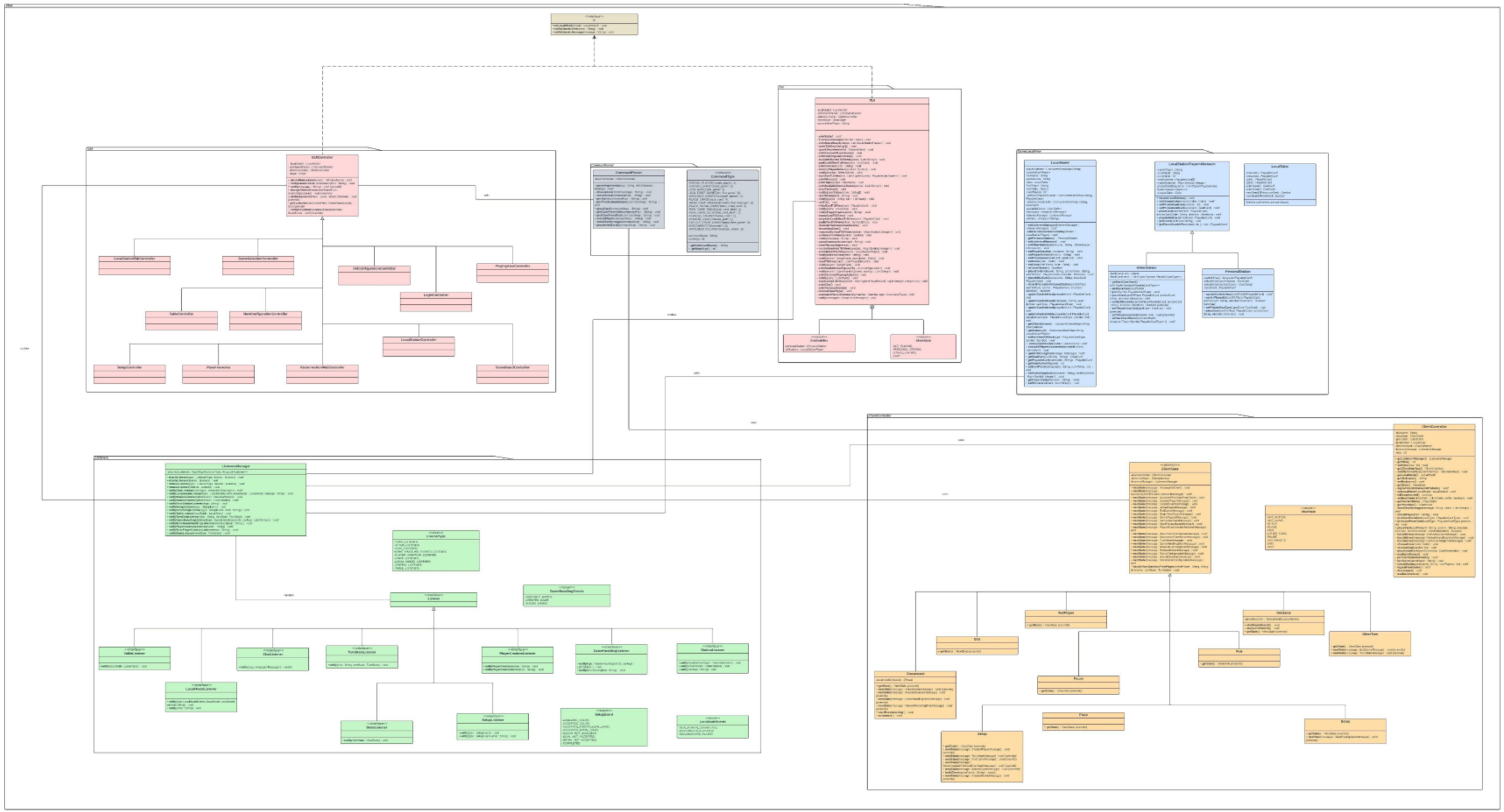
Our Client...


ClientFactory
<<interface>>
ClientFactory
+ clientFactory(controller : ClientController) : ClientInterface
ClientRMIFactory
+ clientFactory(controller : ClientController) : ClientInterface
ClientTCPFactory
+ clientFactory(controller : ClientController) : ClientInterface
Configuration
ConfigurationManager
+ saveConfiguration(configuration Configuration) : void
+ retrieveConfiguration(nick : String) : Configuration
+ retrieveConfiguration() : Configuration
+ deleteConfiguration() : void
+ deleteConfiguration(nick : String) : void
Configuration
- nick : String
- token : String
- timestamp : String
- connectionType : ConnectionType
+ getNick() : String
+ getToken() : String
+ getTimestamp() : String
+ getConnectionType() : ConnectionType
ClientRMI
VirtualClient
+ pushUpdate(MessageToClient message) : void
ClientRMI
- registry : Registry
- virtualMainServer : VirtualMainServer
- virtualGameServer : VirtualGameServer
- virtualGameServerLock : Object
- heartBeatManager : HeartBeatManager
- nickname : String
- messageHandler : MessageHandler
- actionParser : ActionParser
- virtualServerMethodsInvoker : ExecutorService
+ connect(nickname : String) : void
+ createGame(gameName : String, numPlayers : Int) : void
+ createGame(gameName : String, numPlayers : Int, seed : Int) : void
+ joinGame(gameName : String) : void
+ joinFirstAvailableGame() : void
+ reconnect() : void
+ logoutFromGame() : void
+ disconnect() : void
+ signalPossibleNetworkProblem() : void
+ sendHeartBeat() : void
+ startSendingHeartbeat() : void
+ stopSendingHeartbeat() : void
+ stopClient() : void
+ placeCard(cardToInsert : String, anchorCard : String, directionToInsert : Direction, orientation CardOrientation)
+ sendChatMessage(UsersToSend : ArrayList<String>, messageToSend : String) : void
+ placeInitialCard(cardOrientation : CardOrientation) : void
+ pickCardFromTable(card : PlayableCard, position : int) : void
+ pickCardFromDeck(type : PlayableCardType) : void
+ chooseColor(color : Color) : void
+ chosePrivateGoalCard(cardIds : int) : void
+ availableGames() : void
+ getMessageHandler() : MessageHandler
+ getNickname() : String
+ pushUpdate(message : MessageToClient) : void
+ configure(nick : String, token : String)
ClientTCP
ClientTCP
- registry : Registry
- virtualMainServer : VirtualMainServer
- virtualGameServer : VirtualGameServer
- virtualGameServerLock : Object
- heartBeatManager : HeartBeatManager
- nickName : String
- messageHandler : MessageHandler
- actionParser : ActionParser
- virtualServerMethodsInvoker : ExecutorService
+ sendMessage : MessageToServer) : void
+ sendMessageToServer() : void
+ sendMessage(message : MessageToServer) : void
+ finalizeSending() : void
+ stopClient() : void
+ getNickname() : String
+ receiveMessages() : void
+ startSendingHeartBeat() : void
+ stopSendingHeartbeat() : void
+ connect(nickname : String) : void
+ createGame(gameName : String, numPlayers : Int) : void
+ createGame(gameName : String, numPlayers : Int, seed : Int) : void
+ joinGame(gameName : String) : void
+ joinFirstAvailableGame() : void
+ reconnect() : void
+ logoutFromGame() : void
+ disconnect() : void
+ signalPossibleNetworkProblem() : void
+ sendHeartBeat() : void
+ placeCard(cardToInsert : String, anchorCard : String, directionToInsert : Direction, orientation CardOrientation)
+ sendChatMessage(UsersToSend : ArrayList<String>, messageToSend : String) : void
+ placeInitialCard(cardOrientation : CardOrientation) : void
+ pickCardFromTable(card : PlayableCard, position : int) : void
+ pickCardFromDeck(type : PlayableCardType) : void
+ chooseColor(color : Color) : void
+ chosePrivateGoalCard(cardIds : int) : void
+ availableGames() : void
+ getMessageHandler() : MessageHandler
+ configure(nick : String, token : String)
MessageHandler
+ messageToHandle : ArrayDeque<MessageToClient>
+ client : ClientInterface
+ localModel : LocalModel
+ setClient(client : ClientInterface) : void
+ update(message : MessageToClient) : void
+ getMessagesToHandle() : ArrayDeque<MessageToClient>
+ interruptMessageHandler() : void
<<interface>>
ClientInterface
<<interface>>
GameManagementInterface
+ createGame(gameName : String, numPlayers : Int) : void
+ createGame(gameName : String, numPlayers : Int, seed : Int) : void
+ joinGame(gameName : String) : void
+ joinFirstAvailableGame() : void
+ placeCard(cardToInsert : String, anchorCard : String, directionToInsert : Direction, orientation CardOrientation)
+ sendChatMessage(UsersToSend : ArrayList<String>, messageToSend : String) : void
+ placeInitialCard(cardOrientation : CardOrientation) : void
+ pickCardFromTable(card : PlayableCard, position : int) : void
+ pickCardFromDeck(type : PlayableCardType) : void
+ chooseColor(color : Color) : void
+ chosePrivateGoalCard(cardIds : int) : void
+ availableGames() : void
+ getNickname() : String
+ logoutFromGame() : void
<<interface>>
ConfigurableClient
+ configure(nick : String, token : String) : void
NetworkManagement
<<interface>>
NetworkManagementInterface
+ connect(nick : String) : void
+ reconnect() : void
+ disconnect() : void
+ signalPossibleNetworkProblem() : void
+ sendHeartBeat() : void
+ startSendingHeartbeat() : void
+ stopSendingHeartbeat() : void
+ stopClient() : void
HeartBeatManager
+ heartBeat() : void
+ runHeartBeatTesterForServer() : void
+ sendHeartBeat() : void
+ startSendingHeartbeat() : void
+ stopSendingHeartbeat() : void

Factory Pattern



Our view...

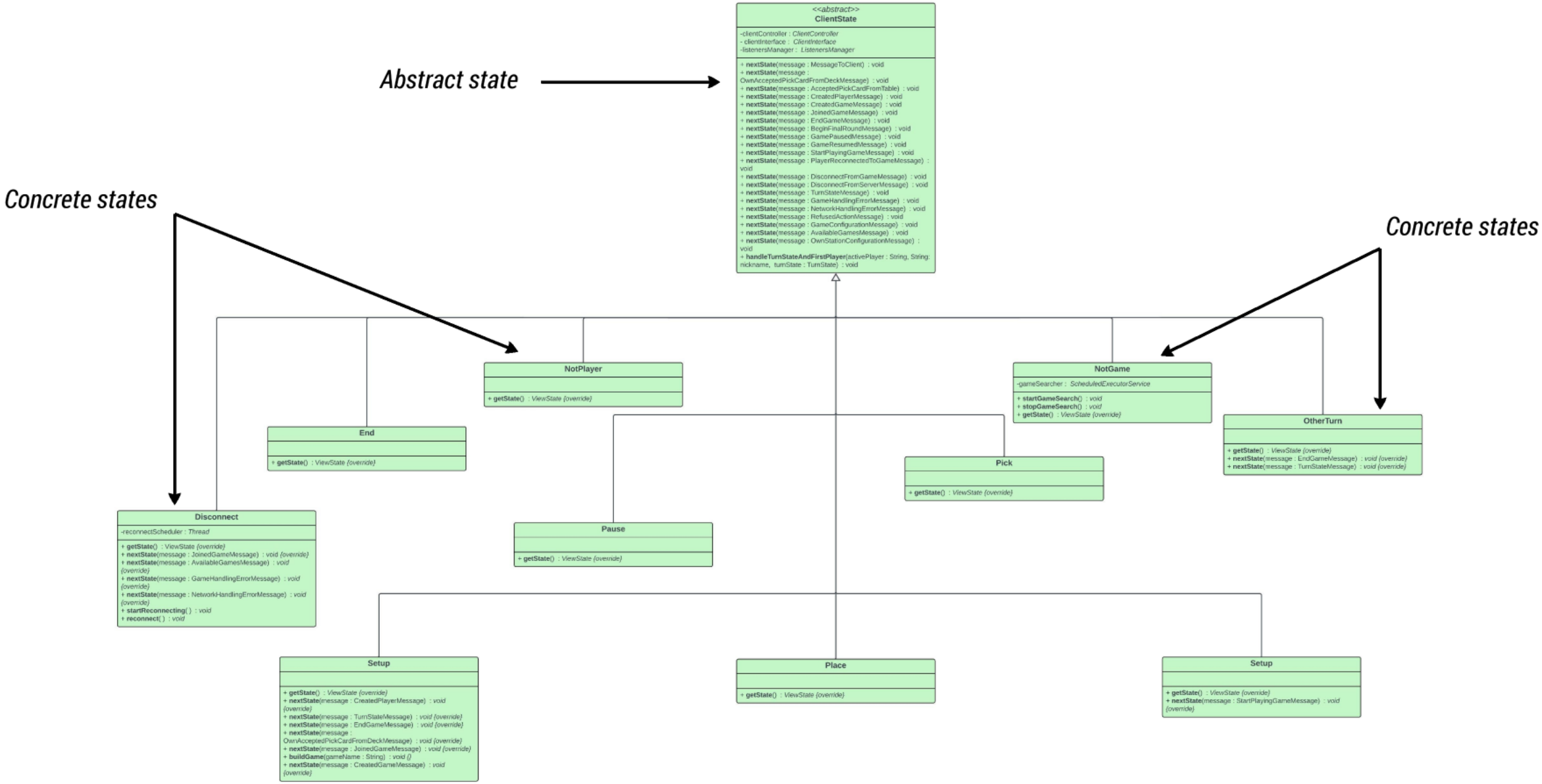


State pattern

Abstract state

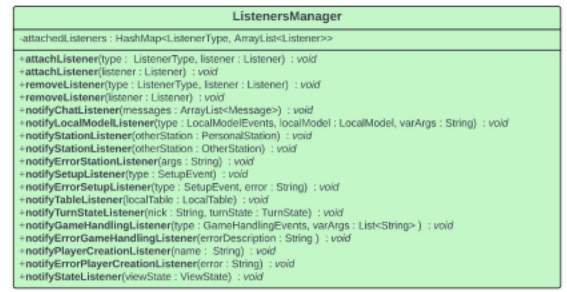
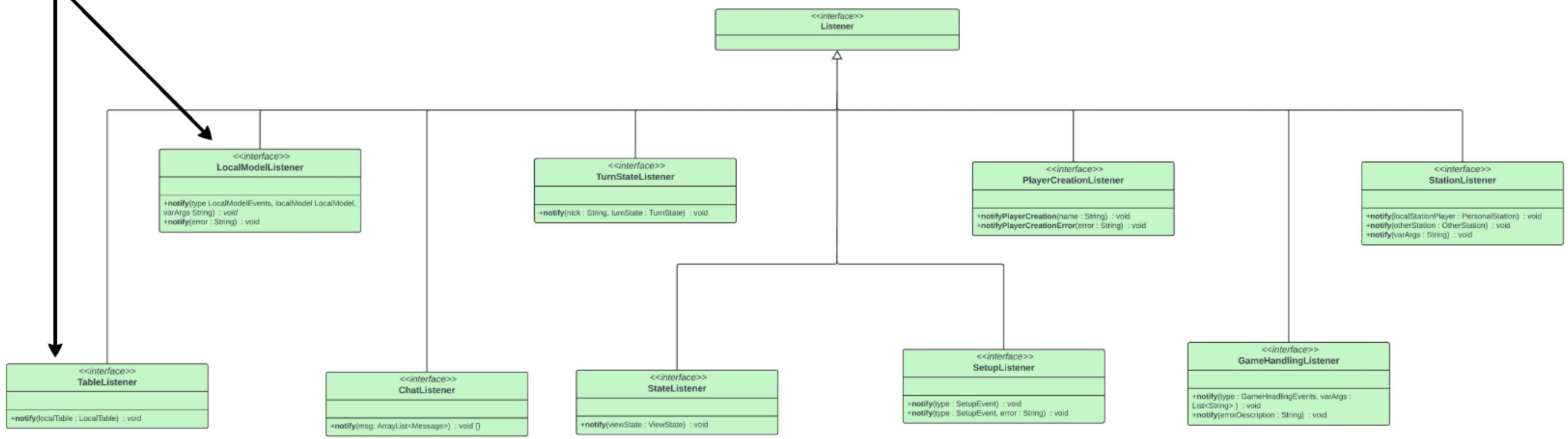
Concrete states

Concrete states



Listener Pattern...

Manager for listeners



Various types of listeners