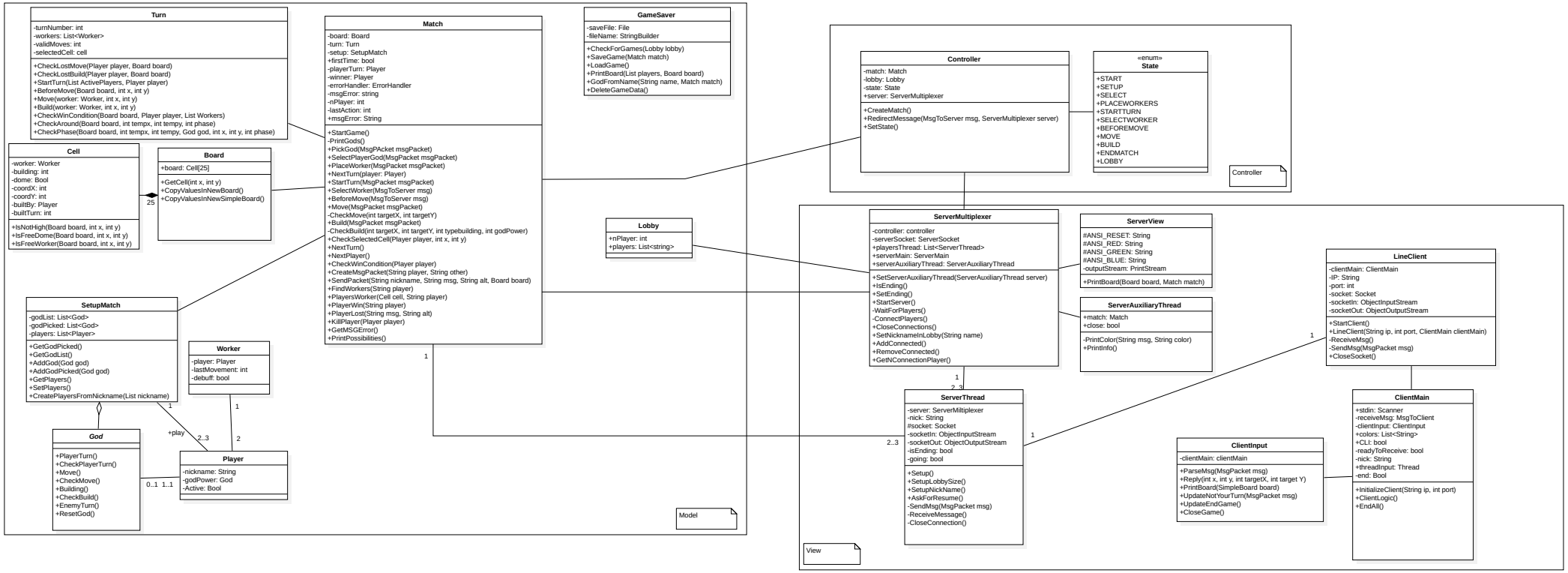
Model::Main


Turn
-turnNumber: int
-workers: List<Worker>
-validMoves: int
-selectedCell: cell
+CheckLostMove(Player player, Board board)
+CheckLostBuild(Player player, Board board)
+StartTurn(List ActivePlayers, Player player)
+BeforeMove(Board board, int x, int y)
+Move(worker: Worker, int x, int y)
+Build(worker: Worker, int x, int y)
+CheckWinCondition(Board board, Player player, List Workers)
+CheckAround(Board board, int tempx, int tempy, int phase)
+CheckPhase(Board board, int tempx, int tempy, God god, int x, int y, int phase)
Match
-board: Board
-turn: Turn
-setup: SetupMatch
+firstTime: bool
-playerTurn: Player
-winner: Player
-errorHandler: ErrorHandler
-msgError: string
-nPlayer: int
-lastAction: int
+msgError: String
+StartGame()
-PrintGods()
+PickGod(MsgPAcket msgPacket)
+SelectPlayerGod(MsgPacket msgPacket)
+PlaceWorker(MsgPacket msgPacket)
+NextTurn(player: Player)
+StartTurn(MsgPacket msgPacket)
+SelectWorker(MsgToServer msg)
+BeforeMove(MsgToServer msg)
+Move(MsgPacket msgPacket)
-CheckMove(int targetX, int targetY)
+Build(MsgPacket msgPacket)
-CheckBuild(int targetX, int targetY, int typebuilding, int godPower)
+CheckSelectedCell(Player player, int x, int y)
+NextTurn()
+NextPlayer()
+CheckWinCondition(Player player)
+CreateMsgPacket(String player, String other)
+SendPacket(String nickname, String msg, String alt, Board board)
+FindWorkers(String player)
+PlayersWorker(Cell cell, String player)
+PlayerWin(String player)
+PlayerLost(String msg, String alt)
+KillPlayer(Player player)
+GetMSGError()
+PrintPossibilities()
GameSaver
-saveFile: File
-fileName: StringBuilder
+CheckForGames(Lobby lobby)
+SaveGame(Match match)
+LoadGame()
+PrintBoard(List players, Board board)
+GodFromName(String name, Match match)
+DeleteGameData()
Cell
-worker: Worker
-building: int
-dome: Bool
-coordX: int
-coordY: int
-builtBy: Player
-builtTurn: int
+IsNotHigh(Board board, int x, int y)
+IsFreeDome(Board board, int x, int y)
+IsFreeWorker(Board board, int x, int y)
Board
+board: Cell[25]
+GetCell(int x, int y)
+CopyValuesInNewBoard()
+CopyValuesInNewSimpleBoard()
25
Lobby
+nPlayer: int
+players: List<string>
SetupMatch
-godList: List<God>
-godPicked: List<God>
-players: List<Player>
+GetGodPicked()
+GetGodList()
+AddGod(God god)
+AddGodPicked(God god)
+GetPlayers()
+SetPlayers()
+CreatePlayersFromNickname(List nickname)
Worker
-player: Player
-lastMovement: int
-debuff: bool
God
+PlayerTurn()
+CheckPlayerTurn()
+Move()
+CheckMove()
+Building()
+CheckBuild()
+EnemyTurn()
+ResetGod()
Player
-nickname: String
-godPower: God
-Active: Bool
+play
2..3
0..1
1..1
1
2
Model
Controller
-match: Match
-lobby: Lobby
-state: State
+server: ServerMultiplexer
+CreateMatch()
+RedirectMessage(MsgToServer msg, ServerMultiplexer server)
+SetState()
«enum»
State
+START
+SETUP
+SELECT
+PLACEWORKERS
+STARTTURN
+SELECTWORKER
+BEFOREMOVE
+MOVE
+BUILD
+ENDMATCH
+LOBBY
Controller
ServerMultiplexer
-controller: controller
-serverSocket: ServerSocket
+playersThread: List<ServerThread>
+serverMain: ServerMain
+serverAuxiliaryThread: ServerAuxiliaryThread
+SetServerAuxiliaryThread(ServerAuxiliaryThread server)
+IsEnding()
+SetEnding()
+StartServer()
-WaitForPlayers()
-ConnectPlayers()
+CloseConnections()
+SetNicknameInLobby(String name)
+AddConnected()
+RemoveConnected()
+GetNConnectionPlayer()
ServerView
#ANSI_RESET: String
#ANSI_RED: String
#ANSI_GREEN: String
#ANSI_BLUE: String
-outputStream: PrintStream
+PrintBoard(Board board, Match match)
ServerAuxiliaryThread
+match: Match
+close: bool
-PrintColor(String msg, String color)
+PrintInfo()
LineClient
-clientMain: ClientMain
-IP: String
-port: int
-socket: Socket
-socketIn: ObjectInputStream
-socketOut: ObjectOutputStream
+StartClient()
+LineClient(String ip, int port, ClientMain clientMain)
-ReceiveMsg()
-SendMsg(MsgPacket msg)
+CloseSocket()
ServerThread
-server: ServerMiltiplexer
-nick: String
#socket: Socket
-socketIn: ObjectInputStream
-socketOut: ObjectOutputStream
-isEnding: bool
-going: bool
+Setup()
+SetupLobbySize()
+SetupNickName()
+AskForResume()
-SendMsg(MsgPacket msg)
-ReceiveMessage()
-CloseConnection()
ClientInput
-clientMain: clientMain
+ParseMsg(MsgPacket msg)
+Reply(int x, int y, int targetX, int target Y)
+PrintBoard(SimpleBoard board)
+UpdateNotYourTurn(MsgPacket msg)
+UpdateEndGame()
+CloseGame()
ClientMain
+stdin: Scanner
-receiveMsg: MsgToClient
-clientInput: ClientInput
+colors: List<String>
+CLI: bool
-readyToReceive: bool
-nick: String
+threadInput: Thread
-end: Bool
+InitializeClient(String ip, int port)
+ClientLogic()
+EndAll()
View