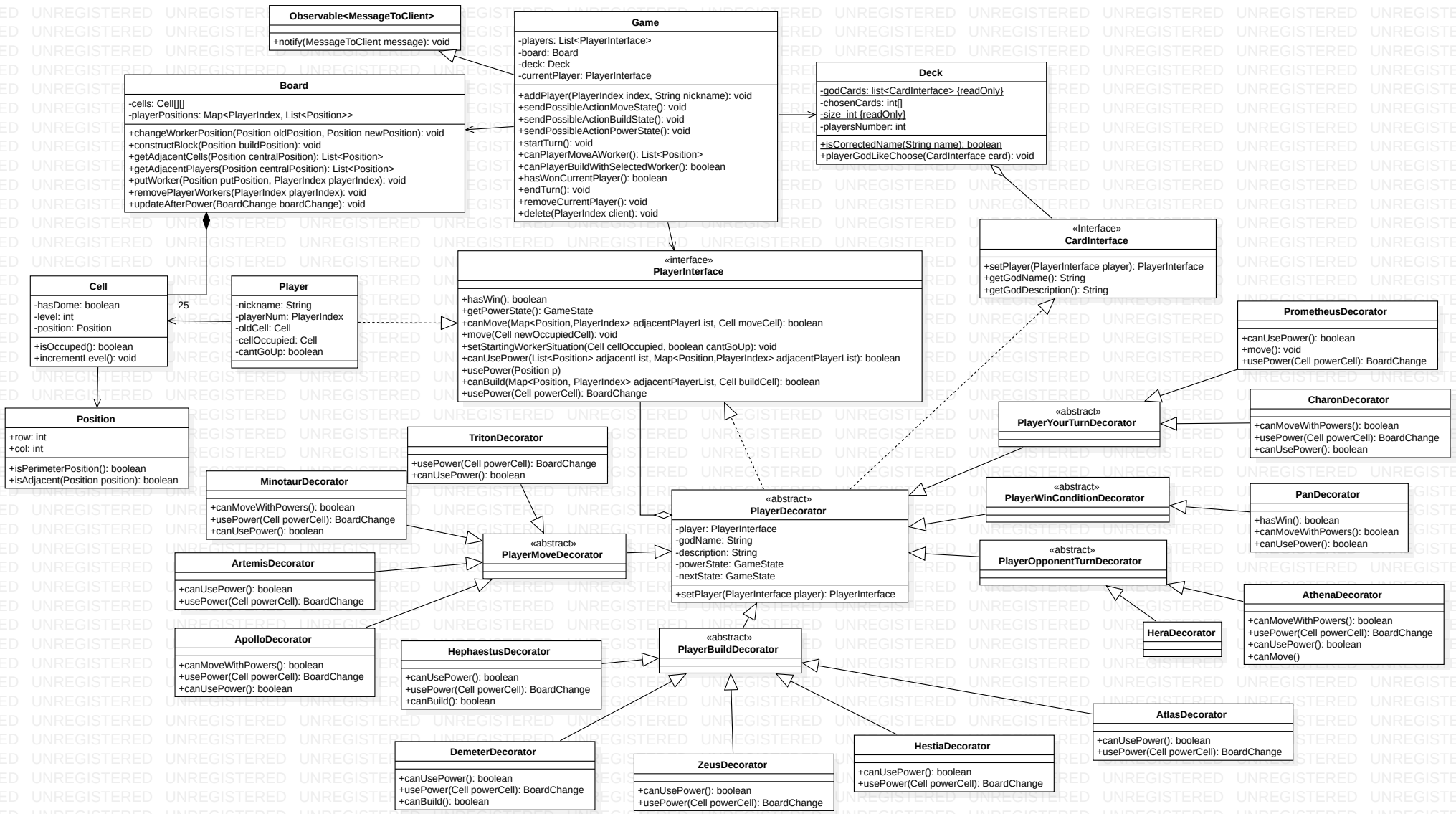
Model.main


Observable<MessageToClient>
+notify(MessageToClient message): void
Game
-players: List<PlayerInterface>
-board: Board
-deck: Deck
-currentPlayer: PlayerInterface
+addPlayer(PlayerIndex index, String nickname): void
+sendPossibleActionMoveState(): void
+sendPossibleActionBuildState(): void
+sendPossibleActionPowerState(): void
+startTurn(): void
+canPlayerMoveAWorker(): List<Position>
+canPlayerBuildWithSelectedWorker(): boolean
+hasWonCurrentPlayer(): boolean
+endTurn(): void
+removeCurrentPlayer(): void
+delete(PlayerIndex client): void
Board
-cells: Cell[][]
-playerPositions: Map<PlayerIndex, List<Position>>
+changeWorkerPosition(Position oldPosition, Position newPosition): void
+constructBlock(Position buildPosition): void
+getAdjacentCells(Position centralPosition): List<Position>
+getAdjacentPlayers(Position centralPosition): List<Position>
+putWorker(Position putPosition, PlayerIndex playerIndex): void
+removePlayerWorkers(PlayerIndex playerIndex): void
+updateAfterPower(BoardChange boardChange): void
Deck
-godCards: list<CardInterface> {readOnly}
-chosenCards: int[]
-size int {readOnly}
-playersNumber: int
+isCorrectedName(String name): boolean
+playerGodLikeChoose(CardInterface card): void
«Interface»
CardInterface
+setPlayer(PlayerInterface player): PlayerInterface
+getGodName(): String
+getGodDescription(): String
Cell
-hasDome: boolean
-level: int
-position: Position
+isOccuped(): boolean
+incrementLevel(): void
25
Player
-nickname: String
-playerNum: PlayerIndex
-oldCell: Cell
-cellOccupied: Cell
-cantGoUp: boolean
«interface»
PlayerInterface
+hasWin(): boolean
+getPowerState(): GameState
+canMove(Map<Position,PlayerIndex> adjacentPlayerList, Cell moveCell): boolean
+move(Cell newOccupiedCell): void
+setStartingWorkerSituation(Cell cellOccupied, boolean cantGoUp): void
+canUsePower(List<Position> adjacentList, Map<Position,PlayerIndex> adjacentPlayerList): boolean
+usePower(Position p)
+canBuild(Map<Position, PlayerIndex> adjacentPlayerList, Cell buildCell): boolean
+usePower(Cell powerCell): BoardChange
PrometheusDecorator
+canUsePower(): boolean
+move(): void
+usePower(Cell powerCell): BoardChange
Position
+row: int
+col: int
+isPerimeterPosition(): boolean
+isAdjacent(Position position): boolean
«abstract»
PlayerYourTurnDecorator
CharonDecorator
+canMoveWithPowers(): boolean
+usePower(Cell powerCell): BoardChange
+canUsePower(): boolean
TritonDecorator
+usePower(Cell powerCell): BoardChange
+canUsePower(): boolean
MinotaurDecorator
+canMoveWithPowers(): boolean
+usePower(Cell powerCell): BoardChange
+canUsePower(): boolean
«abstract»
PlayerDecorator
-player: PlayerInterface
-godName: String
-description: String
-powerState: GameState
-nextState: GameState
+setPlayer(PlayerInterface player): PlayerInterface
«abstract»
PlayerWinConditionDecorator
PanDecorator
+hasWin(): boolean
+canMoveWithPowers(): boolean
+canUsePower(): boolean
«abstract»
PlayerMoveDecorator
ArtemisDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange
«abstract»
PlayerOpponentTurnDecorator
AthenaDecorator
+canMoveWithPowers(): boolean
+usePower(Cell powerCell): BoardChange
+canUsePower(): boolean
+canMove()
HeraDecorator
ApolloDecorator
+canMoveWithPowers(): boolean
+usePower(Cell powerCell): BoardChange
+canUsePower(): boolean
HephaestusDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange
+canBuild(): boolean
«abstract»
PlayerBuildDecorator
AtlasDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange
DemeterDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange
+canBuild(): boolean
ZeusDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange
HestiaDecorator
+canUsePower(): boolean
+usePower(Cell powerCell): BoardChange