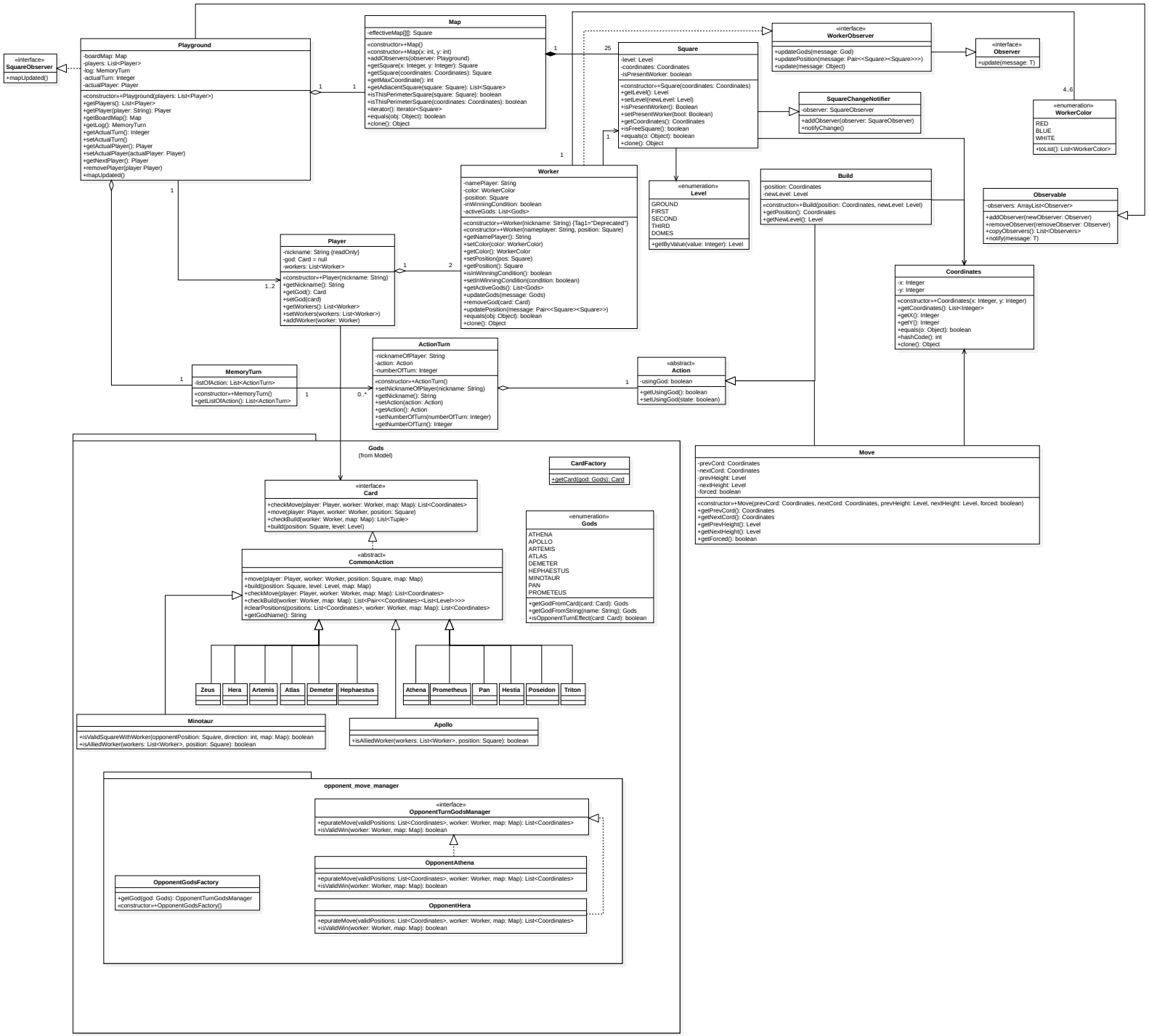
Model::Main


«interface»
SquareObserver
+mapUpdated()
Playground
-boardMap: Map
-players: List<Player>
-log: MemoryTurn
-actualTurn: Integer
-actualPlayer: Player
«constructor»+Playground(players: List<Player>)
+getPlayers(): List<Player>
+getPlayer(player: String): Player
+getBoardMap(): Map
+getLog(): MemoryTurn
+getActualTurn(): Integer
+setActualTurn()
+getActualPlayer(): Player
+setActualPlayer(actualPlayer: Player)
+getNextPlayer(): Player
+removePlayer(player Player)
+mapUpdated()
Map
-effectiveMap[][]: Square
«constructor»+Map()
«constructor»+Map(x: int, y: int)
+addObservers(observer: Playground)
+getSquare(x: Integer, y: Integer): Square
+getSquare(coordinates: Coordinates): Square
+getMaxCoordinate(): int
+getAdiacentSquare(square: Square): List<Square>
+isThisPerimeterSquare(square: Square): boolean
+isThisPerimeterSquare(coordinates: Coordinates): boolean
+iterator(): Iterator<Square>
+equals(obj: Object): boolean
+clone(): Object
«interface»
WorkerObserver
+updateGods(message: God)
+updatePosition(message: Pair<<Square><Square>>>)
+update(message: Object)
«interface»
Observer
+update(message: T)
Square
-level: Level
-coordinates: Coordinates
-isPresentWorker: boolean
«constructor»+Square(coordinates: Coordinates)
+getLevel(): Level
+setLevel(newLevel: Level)
+isPresentWorker(): Boolean
+setPresentWorker(bool: Boolean)
+getCoordinates(): Coordinates
+isFreeSquare(): boolean
+equals(o: Object): boolean
+clone(): Object
SquareChangeNotifier
-observer: SquareObserver
+addObserver(observer: SquareObserver)
+notifyChange()
«enumeration»
WorkerColor
RED
BLUE
WHITE
+toList(): List<WorkerColor>
«enumeration»
Level
GROUND
FIRST
SECOND
THIRD
DOMES
+getByValue(value: Integer): Level
Worker
-namePlayer: String
-color: WorkerColor
-position: Square
-inWinningCondition: boolean
-activeGods: List<Gods>
«constructor»+Worker(nickname: String) {Tag1="Deprecated"}
«constructor»+Worker(nameplayer: String, position: Square)
+getNamePlayer(): String
+setColor(color: WorkerColor)
+getColor(): WorkerColor
+setPosition(pos: Square)
+getPosition(): Square
+isInWinningCondition(): boolean
+setInWinningCondition(condition: boolean)
+getActiveGods(): List<Gods>
+updateGods(message: Gods)
+removeGod(card: Card)
+updatePosition(message: Pair<<Square><Square>>)
+equals(obj: Object): boolean
+clone(): Object
Build
-position: Coordinates
-newLevel: Level
«constructor»+Build(position: Coordinates, newLevel: Level)
+getPosition(): Coordinates
+getNewLevel(): Level
Observable
-observers: ArrayList<Observer>
+addObserver(newObserver: Observer)
+removeObserver(removeObserver: Observer)
+copyObservers(): List<Observers>
+notify(message: T)
Player
-nickname: String {readOnly}
-god: Card = null
-workers: List<Worker>
«constructor»+Player(nickname: String)
+getNickname(): String
+getGod(): Card
+setGod(card)
+getWorkers(): List<Worker>
+setWorkers(workers: List<Worker>)
+addWorker(worker: Worker)
Coordinates
-x: Integer
-y: Integer
«constructor»+Coordinates(x: Integer, y: Integer)
+getCoordinates(): List<Integer>
+getX(): Integer
+getY(): Integer
+equals(o: Object): boolean
+hashCode(): int
+clone(): Object
ActionTurn
-nicknameOfPlayer: String
-action: Action
-numberOfTurn: Integer
«constructor»+ActionTurn()
+setNicknameOfPlayer(nickname: String)
+getNickname(): String
+setAction(action: Action)
+getAction(): Action
+setNumberOfTurn(numberOfTurn: Integer)
+getNumberOfTurn(): Integer
MemoryTurn
-listOfAction: List<ActionTurn>
«constructor»+MemoryTurn()
+getListOfAction(): List<ActionTurn>
«abstract»
Action
-usingGod: boolean
+getUsingGod(): boolean
+setUsingGod(state: boolean)
Move
-prevCord: Coordinates
-nextCord: Coordinates
-prevHeight: Level
-nextHeight: Level
-forced: boolean
«constructor»+Move(prevCord: Coordinates, nextCord: Coordinates, prevHeight: Level, nextHeight: Level, forced: boolean)
+getPrevCord(): Coordinates
+getNextCord(): Coordinates
+getPrevHeight(): Level
+getNextHeight(): Level
+getForced(): boolean
Gods
(from Model)
CardFactory
+getCard(god: Gods): Card
«interface»
Card
+checkMove(player: Player, worker: Worker, map: Map): List<Coordinates>
+move(player: Player, worker: Worker, position: Square)
+checkBuild(worker: Worker, map: Map): List<Tuple>
+build(position: Square, level: Level)
«enumeration»
Gods
ATHENA
APOLLO
ARTEMIS
ATLAS
DEMETER
HEPHAESTUS
MINOTAUR
PAN
PROMETEUS
+getGodFromCard(card: Card): Gods
+getGodFromString(name: String): Gods
+isOpponentTurnEffect(card: Card): boolean
«abstract»
CommonAction
+move(player: Player, worker: Worker, position: Square, map: Map)
+build(position: Square, level: Level, map: Map)
+checkMove(player: Player, worker: Worker, map: Map): List<Coordinates>
+checkBuild(worker: Worker, map: Map): List<Pair<<Coordinates><List<Level>>>>
#clearPositions(positions: List<Coordinates>, worker: Worker, map: Map): List<Coordinates>
+getGodName(): String
Zeus
Hera
Artemis
Atlas
Demeter
Hephaestus
Athena
Prometheus
Pan
Hestia
Poseidon
Triton
Minotaur
+isValidSquareWithWorker(opponentPosition: Square, direction: int, map: Map): boolean
+isAlliedWorker(workers: List<Worker>, position: Square): boolean
Apollo
+isAlliedWorker(workers: List<Worker>, position: Square): boolean
opponent_move_manager
«interface»
OpponentTurnGodsManager
+epurateMove(validPositions: List<Coordinates>, worker: Worker, map: Map): List<Coordinates>
+isValidWin(worker: Worker, map: Map): boolean
OpponentAthena
+epurateMove(validPositions: List<Coordinates>, worker: Worker, map: Map): List<Coordinates>
+isValidWin(worker: Worker, map: Map): boolean
OpponentHera
+epurateMove(validPositions: List<Coordinates>, worker: Worker, map: Map): List<Coordinates>
+isValidWin(worker: Worker, map: Map): boolean
OpponentGodsFactory
+getGod(god: Gods): OpponentTurnGodsManager
«constructor»+OpponentGodsFactory()
1
1
1
25
1
1
1
1
1
1
1..2
2
1
0..*
1
1
4..6