ClientView

Model

<<Enumeration>>
Color
BEIGE
BLUE
WHITE

Player
- NICKNAME : String
- GAME : Game
- god : God
- workers: ArrayList<Worker>
- color: Color
- canWinInPerimeter : boolean
- canMoveUp : boolean
- CLIENT : ClientView

+ Player(game : Game, nickname: String, clientView : ClientView)
+ getNickname():String
+ getGod() : God
+ setGod(god : God) : void
+ getColor() : Color
+ setColor() : void
+ setPermissionToWinInPerimeter(value : boolean) : void
+ setPermissionToMoveUp(value : boolean) : void
+ getCanMoveUp() : boolean
+ getCanWinInPerimeter() : boolean
+ getWorkers() : ArrayList<Worker>
+ getGame() : Game
+ lose() : void
+ getClient() : ClientView

Game
- BOARD: Board
- numberOfPlayers: int
- PLAYERS: ArrayList<Player>
- CHOSENGODS: ArrayList<God>
- challenger : Player

+ Game(numberOfPlayers : int)
+ addPlayer(nickname : String, clientView : ClientView) : Player
- randomChallenger() : void
+ addGodChosenByChallenger(godChosenByChallenger) : void
+ getBoard : Board
+ getNumberOfPlayers() : int
+ getPlayers() : ArrayList<Player>
+ getChallenger() : Player
+ getChosenGods() : ArrayList<God>
+ removePlayer(loser : Player) : void

Board
- board : Cell[][]
- SIDE : int
+ Board()
+ findCell(x : int, y: int) : Cell
+ isInBoard(x : int, y: int) : boolean

Use

Cell
- dome: boolean
- level: int
- worker: Worker
- X : int
- Y : int
- INPERIMETER : boolean
- CELLOBSERVERS : ArrayList<ClientViewObserver>

+ Cell(x : int, y : int)
+ getLevel(): int
+ setLevel(level : int): void
+ setDome(dome : boolean): void
+ setWorker(worker : Worker): void
+ buildBlock(): void
+ hasDome(): boolean
+ buildDome(): void
+ hasWorker(): boolean
+ moveIn(worker : Worker): void
+ moveOut(): void
+ isInPerimeter(): boolean
+ isOccupied(): boolean
+ getWorker(): Worker
+ getX(): int
+ getY(): int
+ register(c : CellObserver): void
+ unregister(c : CellObserver): void
+ notifyObservers() : void

<<Enumerator>>
Sex
MALE
FEMALE

Worker
- PLAYER : Player
- SEX : Sex
- position: Cell
- MOVEMAP : WorkerMoveMap
- BUILDMAP : WorkerBuildMap
- level: int
- levelVariation : int

+ Worker(player : Player, sex : Sex)
+ setPosition(x : int, y : int): void
+ setPosition(newPosition : Cell) : void
+ buildBlock(x : int, y : int) : void
+ buildDome(x : int, y : int) : void
+ getPlayer() : Player
+ getLevel(): int
+ getLevelVariation(): int
+ getPosition(): Cell
+ getSex(): Sex
+ notifyObserver()
+ getMoveMap(): WorkerMoveMap
+ getBuildMap() : WorkerBuildMap
+ swapPosition(newPosition : Cell) : void

Use

<<Abstract Class>>
WorkerMap
- BOARD : Board
- WORKER : Worker
- MATRIX : boolean[][]
- N : int

+ WorkerMap(worker : Worker)
+ getWorker(): Worker
+ domeCellFalse() : void
+ WorkerCellFalse() : void
+ occupiedCellFalse() : void
+ friendlyWorkerCellFalse() : void
- setCenterPosition(center : boolean) : void
- getBooleanCellWorkerMap() : int, j : int) : boolean
- getBooleanCellBoard(i : int, j : int) : boolean
- getAbsolutePosition(i : int, j : int) : Cell
+ levelDifferenceLessEqualThanX(x : int) : void
+ reachNearHigherLevel() : boolean
+ updateCellsOutOfMap() : void
+ neighboringEnemyWorkers() : ArrayList<Worker>
+ anyTrueCell() : boolean
+ resetMap() : void
+ printMap() : void

Extends

Extends

WorkerMoveMap
+ WorkerMoveMap(worker : Worker)
+ cannotMoveInDomeCell() : void
+ cannotMoveInWorkerCell() : void
+ cannotMoveInOccupiedCell() : void
+ cannotMoveInFriendlyWorkerCell() : void
+ cannotStayStill() : void
+ canStayStill() : void
+ isAllowedToMoveBoard(i : int, j : int) : boolean
+ isAllowedToMoveWorkersMap(i : int, j : int) : boolean
+ updateMoveUpRestrictions() : void
+ anyAvailableMovePosition() : boolean
+ anyOneLevelHigherCell() : boolean

WorkerBuildMap
+ WorkerBuildMap(worker : Worker)
+ cannotBuildInOccupiedCell() : void
+ cannotBuildUnderneath() : void
+ canBuildUnderneath() : void
+ isAllowedToBuildBoard(i : int, j : int) : boolean
+ anyAvailableBuildPosition() : boolean

1:1   2:3   1:1   1:1   0:1   1:1