Object Model

GameController

+ playGame(): void
+ initializeGame(): void
+ placeWorkersForPlayer(Player player): void
+ runGameLoop(): void
+ executeMove(Player player, Worker worker): void
+ executeBuild(Player player, Worker worker): void
+ endGame(): void

GameView

- scanner: Scanner

+ getPlayerName(String prompt): String
+ getWorkerPlacement(String playerName, int workerNum): int[ ]
+ getWorkerSelection(): int
+ getPosition(String actionType): int[ ]
+ displayBoard(): void
+ displayWinner(Player winner): void
+ closeScanner(): void

Game

- gameOver: boolean
- winner: Player

+ startGame(Player playerA, Player playerB):void
+ placeWorker(Player player, Worker worker, Cell cell): boolean
+ executeMove(Player player, Worker worker, Cell targetCell): boolean
+ executeBuild(Player player, Worker worker, Cell targetCell): boolean
+ checkWinCondition(): void
+ nextTurn(): void
+ isGameOver(): boolean
+ getWinner(): Player

DemeterBuildStrategy

+ isValidBuild(Game game, Player player, Worker worker, Cell targetCell): boolean
+ afterBuild(Game game, Player player, Worker worker, Cell target): void+ resetTurnState(): void
+ allowsSecondBuild(): boolean
+ wantsToBuildAgain(): boolean

DefaultBuildStrategy

+ isValidBuild(Game game, Player player, Worker worker, Cell targetCell): boolean
+ afterBuild(Game game, Player player, Worker worker, Cell target): void
+ performBuild(Game game, Player player, Worker worker, Cell targetCell, boolean buildDome): void

<<interface>>BuildStrategy

+ isValidBuild(Game game, Player player, Worker worker, Cell targetCell): boolean
+ afterBuild(Game game, Player player, Worker worker, Cell target): void
+ performBuild(Game game, Player player, Worker worker, Cell targetCell, boolean buildDome): void
+ resetTurnState(): void
+ allowsSecondBuild(): boolean
+ wantsToBuildAgain(): boolean

God

-moveStrategy: MoveStrategy
-buildStrategy: BuildStrategy
-winConditionStrategy: WinConditionStrategy

Player

- name: = String
- god: God

+ ownsWorker(Worker worker): boolean
+ getWorkers(): List<Worker>
+ getName(): String

Board

+ isValidPosition(int x, int y): boolean
+ getCell(int x, int y): Cell
+ isAdjacent(Cell cell1, Cell cell2): boolean
+ placeWorker(Worker worker, Cell targetCell): boolean

DefaultMoveStrategy

+ isValidMove(Game game, Player player, Worker worker, Cell targetCell): boolean
+ performMove(Game game, Player player, Worker worker, Cell from, Cell to): void
+ afterMove(Game game, Player player, Worker worker, Cell from, Cell to) void

<<interface>>MoveStrategy

+ isValidMove(Game game, Player player, Worker worker, Cell targetCell): boolean
+ performMove(Game game, Player player, Worker worker, Cell from, Cell to): void
+ afterMove(Game game, Player player, Worker worker, Cell from, Cell to) void

<<enumeration>> Occupancy

EMPTY
WORKER
DOME

DefaultWinConditionStrategy

+ checkWin(Game game, Player player, Worker worker, Cell from, Cell to): boolean

<<interface>>WinConditionStrategy

+ checkWin(Game game, Player player, Worker worker, Cell from, Cell to): boolean

Worker

+ canClimb(int targetLevel): boolean
+ moveTo(Cell targetCell): void
+ buildOn(Cell targetCell): void
+ getPosition(): Cell
+ setPosition(Cell position): void

Cell

- x: int
- y: int
- level: int
- occupancy: Occupancy

+ getLevel( ): int
+ increaseLevel( ): void
+ isOccupied( ): boolean
+ setOccupancy(Occupancy ocuppancy): void