edu.monash

game

actions

<<Interface>> Action
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : void

LoadAction
- - filepath: String
- - deserializer: Deserializer
- + LoadAction(player : Player) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : boolean
- - load() : void

MoveAction
- - move: Move
- + MoveAction(player : Player, from: Integer, to: Integer) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : void

SaveAction
- - filePath : String
- - serializer : Serializer
- + SaveAction(player : Player) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : boolean
- - save() : void

UndoAction
- + UndoAction(player : Player) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : void

PlaceAction
- - move: Move {read only}
- + PlaceAction(player: Player, to: Integer) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : void

RemoveAction
- - move: Move
- + RemoveAction(player : Player, to: Integer) <<constructor>>
- + isValid(game : Game, player : Player) : boolean
- + executeOn(Game game) : void

Deserializer
- - filePath: String
- - load(): String

Serializer
- - filePath : String
- - save(): String

«enumeration» PieceColour
- WHITE
- BLACK

Board
- - NUM_RINGS : int {read only} <<static>>
- - NUM_POSITIONS_PER_RING : int {read only} <<static>>
- - postions : List<Position> {read only}
- + Board() <<constructor>>
- - createBoardStructure() : List<Position>
- + getPosition(id : Integer) : Position
- - previousOfY(y : int) : int
- - nextOfY(y : int) : int
- - previousOfX(x : int) : int
- - nextOfX(x : int) : int
- - toPositiveIndex(index : int, length : int) : int

Position
- - idCounter : int {read only} <<static>>
- - id : int {read only}
- - neighbourUp : Position
- - neighbourDown : Position
- - neighbourLeft : Position
- - neighbourRight : Position
- - occupiedBy : PieceColour
- + Position() <<constructor>>
- + getId() : int
- + withUpNeighbour(neighbourUp : Position): Position
- + withDownNeighbour(neighbourDown : Position): Position
- + withLeftNeighbour(neighbourLeft : Position): Position
- + withRightNeighbour(neighbourRight : Position): Position
- + isNeighbourTo(other : Position): boolean
- + canPiecePlaced(player : Player) : boolean
- + canPieceBeMoved(player : Player) : boolean
- + canPieceBeRemoved(player : Player) : boolean
- + getPiece() : PieceColour
- + setPiece(pieceColour : PieceColour) : void
- + isInHorizontalMill() : boolean
- + isInVerticalMill() : boolean
- + isHorizontalAnchor() : boolean
- + isVerticalAnchor() : boolean

Move
- - piece : PieceColour {read only}
- - from : Integer {read only}
- - to : Integer {read only}
- + Move(pieceColour : PieceColour,from : Integer,to : Integer) <<constructor>>
- + getFrom() : Integer
- + getTo() : Integer
- + executeOn(board : Board) : void

Game
- - board : Board
- - player1 : Player
- - player2 : Player
- - currentPlayer : Player
- - turnCount : int
- - movesPlayed : Stack<Move>
- + Game() <<constructor>>
- + getBoard() : Board
- + execute(action : Action) : boolean
- + storePlayedMove(move : Move) : void
- + initializeNewGame() : void
- + switchActivePlayer() : void
- + getPlayer() : Player
- + getOpponent() : Player
- + isRunning() : boolean

player

<<Interface>> PlayerPhase
- + validate(board : Board, move : Move) : boolean
- + transition(): void

JumpPhase
- - player : Player {read only}
- + JumpPhase(player : Player) <<constructor>>
- + validate(board : Board, move : Move) : boolean
- + transition(): void

SlidePhase
- - player : Player {read only}
- + SlidePhase(player : Player) <<constructor>>
- + validate(board : Board, move : Move) : boolean
- + transition()

PlacePhase
- - player : Player {read only}
- + PlacePhase(player : Player) <<constructor>>
- + validate(board : Board, move : Move) : boolean
- + transition()

Player
- - pieceColour : PieceColour {read only}
- - piecesOnHand: int
- - piecesOnBoard: int
- - hasFormedMill: boolean
- - phase : PlayerPhase
- + Player(pieceColour : PieceColour, initialPhaseConstructor : Function<Player, PlayerPhase>) <<constructor>>
- + getPieceColour() : PieceColour
- + getPiecesOnHand(): int
- + getPiecesOnBoard(): int
- + incrementPiecesOnHand(): void
- + decrementPiecesOnHand(): void
- + incrementPiecesOnBoard(): void
- + decrementPiecesOnBoard(): void
- + hasFormedMill(): boolean
- + setHasFormedMill(hasFormedMill: boolean): void
- + hasLost(): boolean
- + attemptTransitionPhase(): void
- + getPhase() : PlayerPhase
- + setPhase(phaseConstructor : Function<Player, PlayerPhase>) : void
- + canPerformMove(board: Board, move: Move): boolean

ViewController
- - game : Game
- - boardGrid : GameBoardPane
- - blackGrid : PlayerHandGridPane
- - whiteGrid : PlayerHandGridPane
- - initialize() : void

Main
- + start(stage : Stage) : void
- + main(args : String[]) : void <<static>>

GameBoardGridPane
- - boardMapping : Integer[][] {read only}<<static>>
- - game : Game
- + initialize(game: Game) : void
- + getPositionId(x: Integer, y: Integer): Integer <<static>>

MouseEventHandler
- - MouseEventHandler(imageView: ImageView)
- + handle(event: MouseEvent) : void
- - onDragDetectedHandler(event: MouseEvent) : void
- - onMouseClickedHandler(event: MouseEvent) : void

DragEventHandler
- - DragEventHandler(imageView: ImageView)
- + handle(event: DragEvent) : void
- - onDragOverHandler(event: DragEvent): void
- - onDragDroppedHandler(event: DragEvent): void
- - onDragDoneHandler(event: DragEvent): void

PlayerHandGridPane
- - game : Game
- + initialize(game: Game) : void

MouseEventHandler
- - MouseEventHandler(imageView: ImageView)
- + handle(event: MouseEvent) : void
- - onDragDetectedHandler(event: MouseEvent): void

DragEventHandler
- - DragEventHandler(imageView: ImageView)
- + handle(event: DragEvent) : void
- - onDragDoneHandler(event: DragEvent): void

Relationships/labels:
- applies to
- has neighbours
- occupied by
- has
- is played on
- creates
- <<executes on>>
- <<creates>>
- initialize / initialise
-