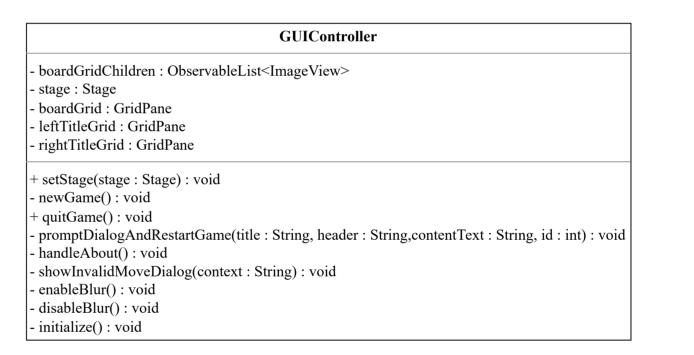
edu.monash game actions has neighbours <<Interface>> applies to Position «enum» Piece + isValid(game : Game, player : Player): boolean - idCounter : int {read only} + excuteOn(Game game): boolean - id : int {read only} neighbourUp : Position **Board** neighbourDown : Position NUM_RINGS : int {read only} - neighbourLeft : Position - neighbourRight : Position NUM_POSITIONS_PER_RING : int {read only} - OccupiedBy : Piece postions : List<Position> {read only} MoveAction SaveAction UndoAction LoadAction + Board() - createBoardStructure() : List<Position> - filePath : String - filepath: String - move : Move {read only} + isValid(game : Game, player : Player): boolean + getId(): int + MoveAction(player : Player, from : Integer, to : Integer) + isValid(game : Game, player : Player): boolean + excuteOn(Game game): boolean - deserializer: Deserialize - serializer : Serializer + withUpNeighbour(neighbourUp : Position): Position + getPosition(id : int) : Position + withDownNeighbour(neighbourDown : Position): Position + isValid(game : Game, player : Player): boolean previousOfY(y:int):int + isValid(game : Game, player : Player): boolean + excuteOn(Game game): boolean + withLeftNeighbour(neighbourLeft : Position): Position nextOfY(y : int) : int + excuteOn(Game game): boolean + excuteOn(Game game): boolean + withRightNeighbour(neighbourRight : Position): Position + canPieceBePlaced() : boolean previousOfX(x : int) : int - load() : void save(): void nextOfX(x : int) : int + canPieceBeMoved(player : Player) : boolean + canPieceBeRemoved(player : Player) : boolean - toPositiveIndex(index : int, length : int) : int creates creates + getPiece() : Piece + setPiece(piece : Piece) : void + isInHorizontalMill(): boolean + isInVerticalMill(): boolean Deserializer Serializer + isHoriziontalAnchor() : boolean + isVerticalAnchor() : boolean - filePath: String - filePath: String - load(): String - save(): String executes on is played on 0...* Move Game - piece : Piece {read only} - board : Board - player1 : Player - from : Integer {read only} - player2 : Player - to : Integer {read only} - currentPlayer : Player + Move(piece : Piece,from : Integer,to : Integer) - trunCount : int + getFrom() : Integer - movesPlayed : Stack<Move> + getTo(): Integer + Game() + executeOn(board : Board) : void + getBoard() : Board + excute(action : Action) : void creates + storePlayerMove(move : Move) : void + initializeNewGame(): void + switchPlayer(): void + getCurrentPlayer() : Player player has PlayerPhase pieceColour : Piece {read only} validate(board : Board, move : Move): Boolear - phase : PlayerPhase has + Player(pieceColour : Piece, initialPhaseConstructor : Function<Player, PlayerPhase>) + getPieceColour() : Piece + getPhase() : PlayerPhase + setPhase(phaseConstructor : Function<Player, PlayerPhase>) : void JumpPhase SlidePhase PlacePhase - player : Player {read only} - player : Player {read only} player : Player {read only} + SlidePhase(player : Player) PlacePhase(player : Player) + JumpPhase(player : Player) · validate(board : Board, move : Move): boolean + validate(board : Board, move : Move): boolean + validate(board : Board, move : Move): boolean

ViewController • GRID_PANE_CELL__FORMAT : DateFormat = new DateFormat(...) {read only} game : Game - boardGridChildren : ObservableList<ImageView> - stage : Stage - boardGrid : GridPane - blackGrid : GridPane whiteGrid : GridPane + ViewController() + setStage(stage : Stage) : void - newGame() : void + quitGame() : void + showPromptDialog(title: String, header: String, content: String, yesCallBack: Runnable, noCallBack: Runnable): void - boardMapping : Integer[][] = {{...}} {read only} + getPositionID(x: int, y: int) : Integer - setUpDragAndDrop(gridPane : GridPane) : void onDragDetectedHandler(child : Node, event : MouseEvent) : void onDragOverHandler(child : Node, event : MouseEvent) : void - onDragDroppedHandler(child : Node, event : MouseEvent) : void enableBlur(): void - disableBlur() : void - initialize() : void



Main stage : Stage) : void

+ start(stage : Stage) : void + main(args : String[]) : void