edu.monash

game

actions

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

LoadAction
- + isValid(Game game, Player player): boolean
- + excuteOn(Game game): boolean

MoveAction
- - move : Move
- + isValid(Game game, Player player): boolean
- + excuteOn(Game game): boolean
- + MoveAction(Player player, Integer from, Integer to)

SaveAction
- + isValid(Game game, Player player): boolean
- + excuteOn(Game game): boolean

UndoAction
- + isValid(Game game, Player player): boolean
- + excuteOn(Game game): boolean

Deserializer

Serializer

applies to

«enum» Piece

has neighbours

Position
- - idCounter : int
- - id : int
- - neighbourUp : Position
- - neighbourDown : Position
- - neighbourLeft : Position
- - neighbourRight : Position
- - OccupiedBy : Piece
- + withUpNeighbour(Position neighbourUp): Position
- + withDownNeighbour(Position neighbourUp): Position
- + withLeftNeighbour(Position neighbourUp): Position
- + withRightNeighbour(Position neighbourUp): Position
- + canPieceBePlaced() : boolean
- + canPieceBeMoved(Player player) : boolean
- + canPieceBeRemoved(Player player) : boolean
- + getPiece() : Piece
- + setPiece(Piece piece) : void
- + isInHorizontalMill() : boolean
- + isInVerticalMill() : boolean
- + isHorizontalAnchor() : boolean
- + isVerticalAnchor() : boolean

2...4

Board
- - NUM_RINGS : int
- - NUM_POSITIONS_PER_RING : int
- - postions : List<Position>
- + Board()
- + getPosition(int id) : Position
- - createBoardStructure() : List<Position>
- - previousOfY(int y) : int
- - nextOfY(int y) : int
- - previousOfX(int x) : int
- - nextOfX(int x) : int
- - toPositiveIndex(int index, int length) : int

has

24

executes on

is played on

Move
- - piece : Piece
- - from : Integer
- - to : Integer
- + Move(Piece piece, Integer from, Integer to)
- + getFrom() : Integer
- + getTo() : Integer
- + executeOn(Board board) : void

0...*

Game
- - board : Board
- - player1 : Player
- - player2 : Player
- - currentPlayer : Player
- - trunCount : int
- - movesPlayed : Stack<Move>
- + Game()
- + getBoard() : Board
- + excute(Action action) : void
- + storePlayerMove(Move move) : void
- + initializeNewGame() : void
- + switchPlayer() : void

creates

player

<<Interface>>
PlayerPhase
- + validate(Board board, Move move): Boolean

has

Player
- - pieceColour : Piece
- - phase : PlayerPhase
- + Player(Piece pieceColour, Function<Player,PlayerPhase> initialPhaseConstructor)
- + getPieceColour() : Piece
- + getPhase() : PlayerPhase
- + setPhase(Function<Player,PlayerPhase> phaseConstructor) : void

2

has

JumpPhase
- - player : Player
- + JumpPhase(Player player)
- + validate(Board board, Move move): Boolean

SlidePhase
- - player : Player
- + SlidePhase(Player player)
- + validate(Board board, Move move): Boolean

PlacePhase
- - player : Player
- + PlacePhase(Player player)
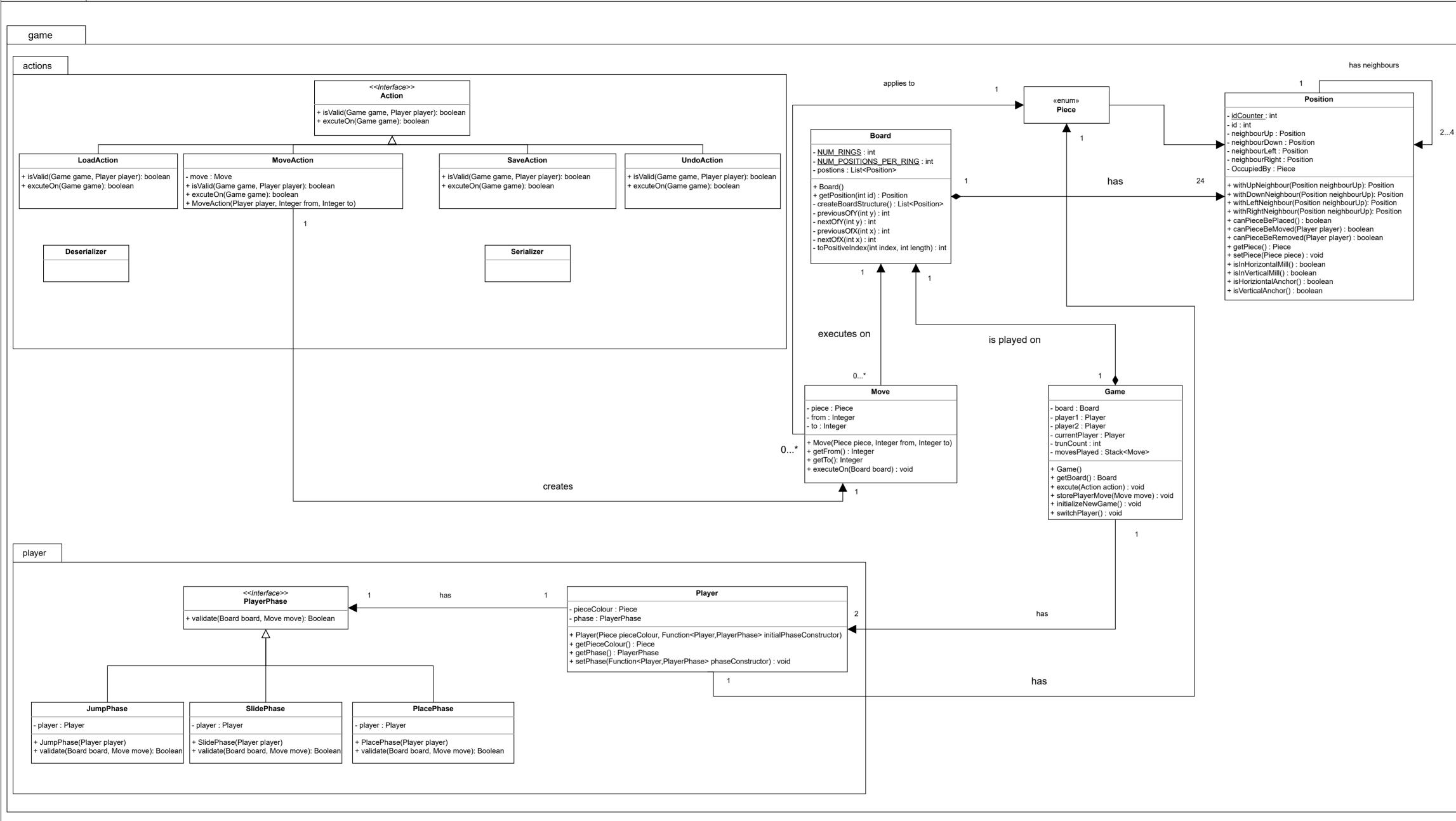- + validate(Board board, Move move): Boolean

GUIController
- - boardGridChildren : ObservableList<ImageView>
- - stage : Stage
- - boardGrid : GridPane
- - leftTitleGrid : GridPane
- - rightTitleGrid : GridPane
- + setStage(Stage stage) : void
- - newGame() : void
- + quitGame() : void
- - promptDialogAndRestartGame(String title, String header, String context, ind id) : void
- - handleAbout() : void
- - showInvalidMoveDialog(String context) : void
- - enableBlur() : void
- - disableBlur() : void
- - initialize() : void

ViewController
- - game : Game
- - boardMapping : Integer[][]
- + ViewController()
- + getPositionId(int x, int y) : Integer

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