CS246 Chess Program UML

Leo Feng, Richard Yuan, Rick Lyu | December 2, 2021

«enumeration»
GAMESTATUS

NO_CHECK
BLACK_IN_CHECK
WHITE_IN_CHECK
BLACK_IS_MATED
WHITE_IS_MATED
STALEMATE

ChessGame

-chessBoard: Board
-players: Player[]
-turn: COLOUR
-gameStatus: GAMESTATUS
-moves: vector<Move>

+init(): void
+checkMoves() : vector<Move>
+makeMove(move : Move) : Boolean

Board

-cells: Cell[][]

+getCell(r : int, c : int) : Cell
+setCell(r : int, c : int, p : Piece) : Cell
+setup(custom : Boolean) : void
+isValidSetup() : Boolean

Subject

+attach(obs : Observer) : void
+detach(obs : Observer) : void
#notifyObservers() : void

Observer

+notify() : void

Text

-boardState: Board

+notify() : void

Graphics

-boardState: Board
-window: Xwindow

+notify() : void

Cell

-row : int
-column : int
-piece : Piece

+isEmpty(): Boolean

Player

-colour: COLOUR
-isComp : Boolean

Move

-fromCell: Cell
-toCell: Cell
-piece: Piece
-capturedPiece: Piece
-castle: Boolean = false

Piece

-captured: Boolean = false
-colour: COLOUR
-moved: Boolean = false
-pieceType: PIECETYPE

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

«enumeration»
PIECETYPE

PAWN
KNIGHT
BISHOP
ROOK
QUEEN
KING

XWindow

Implementation
From A4Q3

«enumeration»
COLOUR

BLACK
WHITE

overload operator=

ComputerPlayer

-level: int

HumanPlayer

Potentially
implement a
Strategy Design
Pattern

King

-castled: Boolean = false

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>
+castleMoves(): Boolean

Queen

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

Rook

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

Bishop

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

Knight

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

Pawn

+validMoves(): vector<Cell>
+checkMoves(): vector<Cell>
+captureMoves(): vector<Cell>
+safeMoves(): vector<Cell>

each function here will take in a Board as an argument

1
64
1
1
2..*
*
*
*