Chess UML

Group: Harsh Patel (h334pate), Jaimil Dalwadi (jdalwadi), Mayank Mehra (m4mehra) | December 6, 2022

Move

+ startRow : int
+ startCol : int
+ endRow : int
+ endCol : int
+ type : MoveType
+ movedPiece : Piece
+ capturedPiece : Piece
+ promote : PieceType

+ setMoveType(MoveType) : void
+ printMove() : void

Player

colour : Colour
board : ChessBoard

+ makeMove(ChessBoard, Colour, String): Move

Subject

- observers: vector<Observer>

+ attach(Observer) : void
+ detach(Observer) : void
+ notifyObservers() : void
+ pieceAt(int, int) : Piece
+ sendMessage(string) : void

Observer

+ notify() : void
+ notifyMessage(string) : void

<<enumeration>> PieceType

King
Queen
Bishop
Knight
Rook
Pawn
NoPiece

<<enumeration>> Colour

White
Black
NoColor

<<enumeration>> MoveType

Normal
Promotion
Enpassent
PawnUp2
Castling
Capture

NPC

- level : Level

+ makeMove(String): Move

Human

+ makeMove(String): Move

AddText

- board : ChessBoard

+ notify() : void
+ notifyMessage(string) : void

AddGraphics

- width : int
- height : int
- subject : ChessBoard
- w : Xwindow

+ notify() : void
+ notifyMessage(string) : void
+ displayBasicBoard() : void

ChessBoard

- pieces : vector<vector<Piece>>
- lastMove : Move
- pWhite : Player
- pBlack : Player
- scoreBoard : ScoreBoard
- isSetup : bool
- gameOn : bool
- turn : int

+ initBasicBoard() : void
+ getGameOn() : bool
+ setupMove() : bool
+ setGameOn() : bool
+ isWhiteMove() : bool
+ gotGameInput(string) : void
+ startGame(string, string) : void
+ render() : void
+ terminateGame(Colour) : void
+ endGameCheck(Colour) : bool
+ applyMove(Move) : void
+ canApplyMove(Move) : bool
+ checkIfInCheck(Colour) : bool
+ checkIfInCheckmate(Colour) : bool
+ putsSelfInCheck(Move, Piece, Move) : bool
+ getLegalMoves(Colour) : vector<Move>
+ canCapture(Colour, int, int) : bool
+ isTurn(Colour) : bool
+ pieceAt(Colour, int, int) : Piece
+ setPieceAt(Colour, int, int, PieceType) : void
+ printScore() : void
+ enterSetup() : void
+ getState() : vector<vector<Piece>>
+ convertToPiece(char) : PieceType
+ resetBoard() : void
+ softClone() : ChessBoard

Xwindow

+ fillRectangle(int, int, int, int, int) : void
+ drawString(int, int, string) : void

Level

+ makeMove(ChessBoard, Colour, String): Move

Level1

+ makeMove(ChessBoard, Colour, String): Move

Level2

+ makeMove(ChessBoard, Colour, String): Move

Level3

+ makeMove(ChessBoard, Colour, String): Move

Level4

+ makeMove(ChessBoard, Colour, String): Move

Scoreboard

- whiteWins: int
- blackWins: int
- stales: int

+ printScore() : void
+ addScore(Colour) : void
+ blackScore() : int
+ whiteScore() : int
+ draws() : int

Piece

colour : Colour
row : int
col : int
weight : int

+ printpiece() : void
+ getWeight() : int
+ getPos() : tuple<int, int>
+ convertChar(char) : PieceType
+ updatePos(int, int) : void
+ isInBoundary(Move) : bool
+ setPiece(PieceType, Colour) : void
+ getColour() : Colour

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

King

- canCastle : bool

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

Queen

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

Bishop

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

Knight

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

Rook

- canCastle : bool

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void

Pawn

+ validMove(Move, vector<vector<Piece>>, int) : bool
+ getPiece() : PieceType
+ clonePiece() : Piece
+ targetedTiles(vector<vector<Piece>>, int) : vector<Move>
+ ableToCastle() : bool
+ setCastle(bool) : void