TextDisplay

+ notify(board: Board, srcRow: int, srcCol: int, destRow: int, destCol: int): void

GraphicsDisplay

+ notify(board: Board, srcRow: int, srcCol: int, destRow: int, destCol: int): void

View

+ *notify(board: Board, srcRow: int, srcCol: int, destRow: int, destCol: int): void*

Pawn

+ getPossibleMoves(): vector<Position>
+ displayChar(): char
+ setMove(): void
+ setEnpassantTrue(): void
+ setEnpassantFalse(): void
+ setToggle(): void
+ getEnpassant(): bool
+ getMoved(): bool
+ getToggle(): bool

Knight

+ getPossibleMoves(): vector<Position>
+ displayChar(): char

Bishop

+ getPossibleMoves(): vector<Position>
+ displayChar(): char

Rock

+ setMove(): void
+ getMove(): bool
+ getPossibleMoves(): vector<Position>
+ displayChar(): char

King

+ getPossibleMoves(): vector<Position>
+ displayChar(): char

Queen

+ getPossibleMoves(): vector<Position>
+ displayChar(): char

Board

+ isInCheckAfterMove( ) : boolean
+ standardBoardSetup() : void
+ switchTurns() : void
+ getTurn() : int
+ makeMove(p: Piece*, pos: Position) : void
+ undoMove(dup: Piece*, captured: bool, startPos: Position, endPos: Position): void
+ duplicate(p: Piece*): Piece*
+ addPiece(type: char, pos: Position): void
+ removePiece(pos: Position): void
+ findKingPosition(colour: string): Position
+ isCheck(colour: string): bool
+ isInCheckAfterMove(currPos: Position, newPos: Position, colour: string): bool
+ getPositionCausingCheck(playerColour: string): Position
+ isStalemate(playerColour: string): bool
+ canBlock(attacker: Position, target: Position, playerColour: string): bool
+ canCapture(attacker: Position, target: Position): bool
+ isCheckmate(playerColour: string): bool
+ getState(): vector<vector<Piece*>>
+ notifyObservers(srcRow: int, srcCol: int, destRow: int, destCol: int): void
+ attach(o: View*): void
+ clearBoard(): void

Piece

+ getX(): int
+ getY(): int
+ setPosition(position: Position): void
+ getColour(): string
+ getType(): Type
+ getScoreValue(): int
+ *getPossibleMoves(): vector<Position>*
+ isValid(newPos: Position): bool
+ *displayChar(): char*

<<enumeration>> Type

- Pawn
- Knight
- Bishop
- Rook
- Queen
- King

Controller

+ convertCoords(coords: string): Position
+ displayScore(): void
+ run(): void
+ setupMode(player1: Player*, player2: Player*): void
+ processSetupCommand(command: string, player1: Player*, player2: Player*): void
+ makeHumanMove(playerColor: string, player: Player*): void
+ makeComputerMove(playerColour: string, player: Player*): void

Player

+ setColour(newColour: string): void
+ getColour(): string
+ getPlayerType(): string
+ addToLostPieces(lostPiece: Piece*): void
+ getLostPieces(): vector<Piece*>
+ getAvailableMoves(): map<Position, vector<Position>>
+ renderAvailableMoves(board: Board*): void
+ printAvailableMoves(): void
+ hasPlacedKing(): bool
+ addPieceType(piece: char): bool
+ removePieceType(piece: Piece::Type): void
+ getComputerLevel(): int

Computer

+ *algorithm(board: Board*): pair<Position, Position>*

Human

LevelOne

+ *algorithm(board: Board*): pair<Position, Position>*
+ *getComputerLevel(): int*

LevelTwo

+ algorithm(board: Board*): pair<Position, Position>
+ getComputerLevel(): int
+ evaluateMove(move: pair<Position, Position>, board: Board*): int

LevelThree

+ algorithm(board: Board*): pair<Position, Position>
+ getComputerLevel(): int
+ evaluateMove(move: pair<Position, Position>, board: Board*): int
+ avoidCapture(move: pair<Position, Position>, board: Board*, playerColour: string): bool

LevelFour

+ algorithm(board: Board*): pair<Position, Position>
+ getComputerLevel(): int
+ evaluateMove(move: pair<Position, Position>, board: Board*): int
+ moveAvoidsCapture(move: pair<Position, Position>, board: Board*, playerColour: string): bool

LevelFive

+ algorithm(board: Board*): pair<Position, Position>
+ loadOpeningBook(): void
+ selectOpeningMove(board: Board*): pair<Position, Position>
+ getComputerLevel(): int