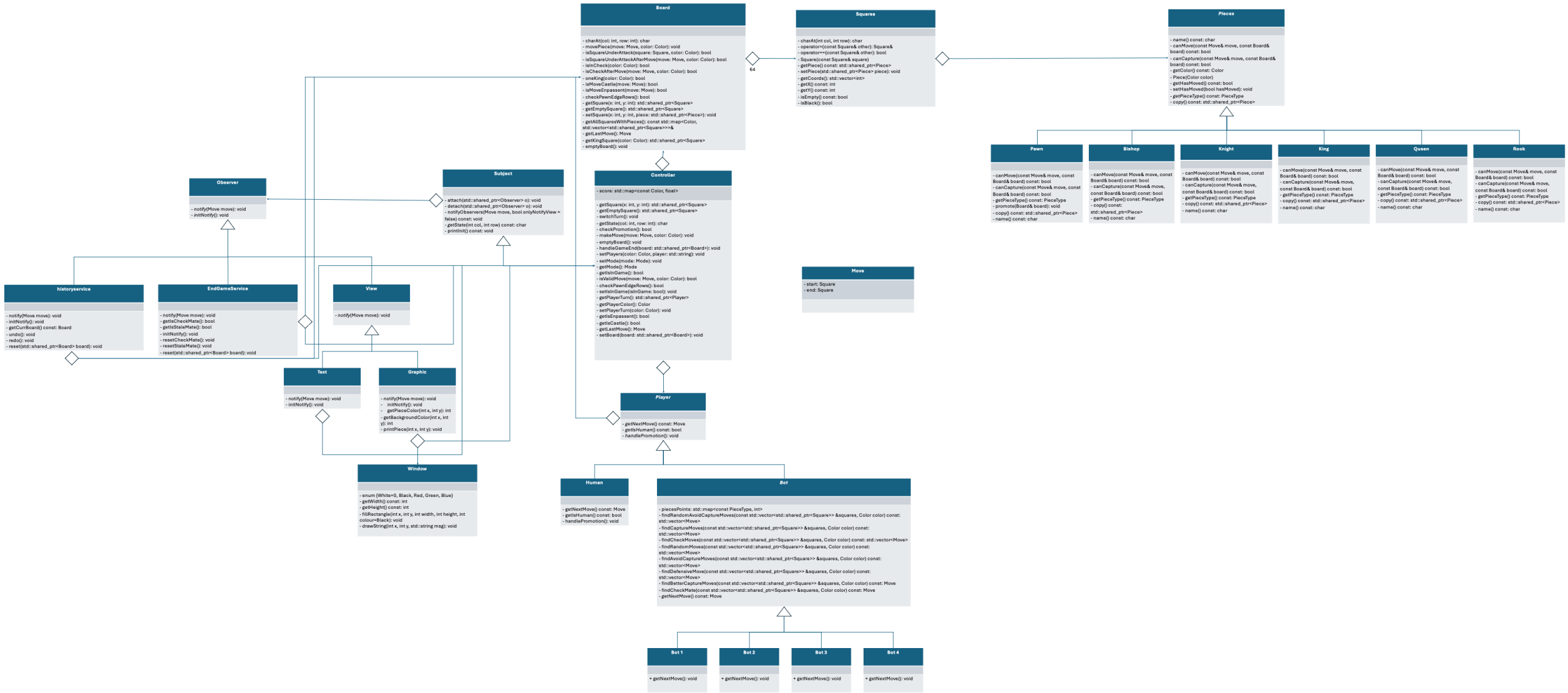
Board
- charAt(col: int, row: int): char
- movePiece(move: Move, color: Color): void
- isSquareUnderAttack(square: Square, color: Color): bool
- isSquareUnderAttackAfterMove(move: Move, color: Color): bool
- isInCheck(color: Color): bool
- isCheckAfterMove(move: Move, color: Color): bool
- oneKing(color: Color): bool
- isMoveCastle(move: Move): bool
- isMoveEnpassant(move: Move): bool
- checkPawnEdgeRows(): bool
- getSquare(x: int, y: int): std::shared_ptr<Square>
- getEmptySquare(): std::shared_ptr<Square>
- setSquare(x: int, y: int, piece: std::shared_ptr<Piece>): void
- getAllSquaresWithPieces(): const std::map<Color, std::vector<std::shared_ptr<Square>>>&
- getLastMove(): Move
- getKingSquare(color: Color): std::shared_ptr<Square>
- emptyBoard(): void
64
Squares
- charAt(int coi, int row): char
- operator=(const Square& other): Square&
- operator==(const Square& other): bool
- Square(const Square& square)
- getPiece() const: std::shared_ptr<Piece>
- setPiece(std::shared_ptr<Piece> piece): void
- getCoords(): std::vector<int>
- getX() const: int
- getY() const: int
- isEmpty() const: bool
- isBlack(): bool
Pieces
- name() const: char
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getColor() const: Color
- Piece(Color color)
- getHasMoved() const: bool
- setHasMoved(bool hasMoved): void
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
Pawn
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- promote(Board& board): void
- copy() const: std::shared_ptr<Piece>
- name() const: char
Bishop
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
- name() const: char
Knight
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
- name() const: char
King
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
- name() const: char
Queen
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
- name() const: char
Rook
- canMove(const Move& move, const Board& board) const: bool
- canCapture(const Move& move, const Board& board) const: bool
- getPieceType() const: PieceType
- copy() const: std::shared_ptr<Piece>
- name() const: char
Observer
- notify(Move move): void
- initNotify(): void
Subject
- attach(std::shared_ptr<Observer> o): void
- detach(std::shared_ptr<Observer> o): void
- notifyObservers(Move move, bool onlyNotifyView = false) const: void
- getState(int coi, int row) const: char
- printInit() const: void
Controller
- score: std::map<const Color, float>
- getSquare(x: int, y: int): std::shared_ptr<Square>
- getEmptySquare(): std::shared_ptr<Square>
- switchTurn(): void
- getState(coi: int, row: int): char
- checkPromotion(): bool
- makeMove(move: Move, color: Color): void
- emptyBoard(): void
- handleGameEnd(board: std::shared_ptr<Board>): void
- setPlayers(color: Color, player: std::string): void
- setMode(mode: Mode): void
- getMode(): Mode
- getIsInGame(): bool
- isValidMove(move: Move, color: Color): bool
- checkPawnEdgeRows(): bool
- setIsInGame(isInGame: bool): void
- getPlayerTurn(): std::shared_ptr<Player>
- getPlayerColor(): Color
- setPlayerTurn(color: Color): void
- getIsEnpassant(): bool
- getIsCastle(): bool
- getLastMove(): Move
- setBoard(board: std::shared_ptr<Board>): void
Move
- start: Square
- end: Square
historyservice
- notify(Move move): void
- initNotify(): void
- getCurrBoard() const: Board
- undo(): void
- redo(): void
- reset(std::shared_ptr<Board> board): void
EndGameService
- notify(Move move): void
- getIsCheckMate(): bool
- getIsStaleMate(): bool
- initNotify(): void
- resetCheckMate(): void
- resetStaleMate(): void
- reset(std::shared_ptr<Board> board): void
View
- notify(Move move): void
Text
- notify(Move move): void
- initNotify(): void
Graphic
- notify(Move move): void
- initNotify(): void
- getPieceColor(int x, int y): int
- getBackgroundColor(int x, int y): int
- printPiece(int x, int y): void
Window
- enum {White=0, Black, Red, Green, Blue}
- getWidth() const: int
- getHeight() const: int
- fillRectangle(int x, int y, int width, int height, int colour=Black): void
- drawString(int x, int y, std::string msg): void
Player
- getNextMove() const: Move
- getIsHuman() const: bool
- handlePromotion(): void
Human
- getNextMove() const: Move
- getIsHuman() const: bool
- handlePromotion(): void
Bot
- piecesPoints: std::map<const PieceType, int>
- findRandomAvoidCaptureMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findCaptureMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findCheckMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findRandomMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findAvoidCaptureMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findDefensiveMove(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: std::vector<Move>
- findBetterCaptureMoves(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: Move
- findCheckMate(const std::vector<std::shared_ptr<Square>> &squares, Color color) const: Move
- getNextMove() const: Move
Bot 1
+ getNextMove(): void
Bot 2
+ getNextMove(): void
Bot 3
+ getNextMove(): void
Bot 4
+ getNextMove(): void