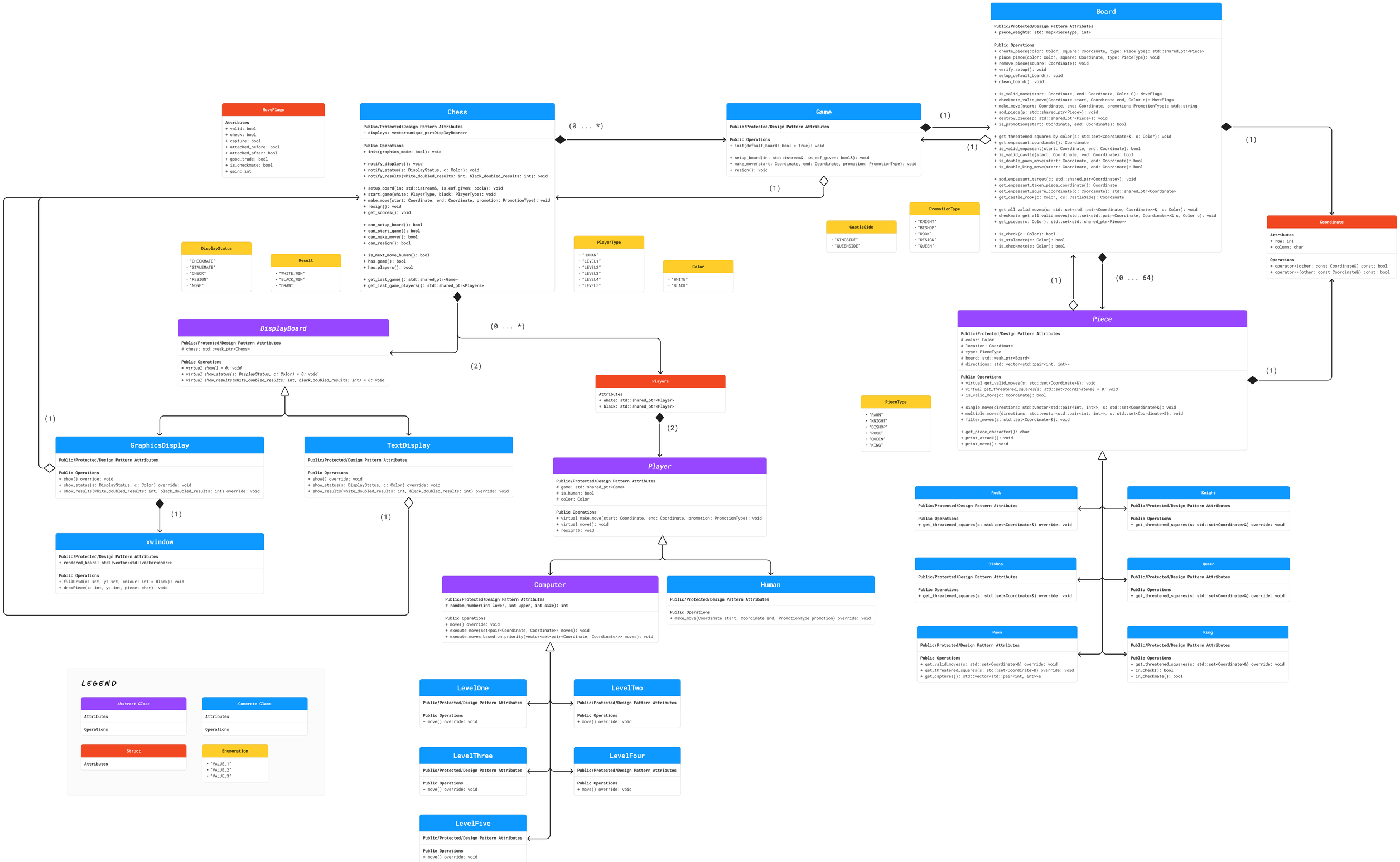
MoveFlags
Attributes
+ valid: bool
+ check: bool
+ capture: bool
+ attacked_before: bool
+ attacked_after: bool
+ good_trade: bool
+ is_checkmate: bool
+ gain: int
Chess
Public/Protected/Design Pattern Attributes
- displays: vector<unique_ptr<DisplayBoard>>
Public Operations
+ init(graphics_mode: bool): void
+ notify_displays(): void
+ notify_status(s: DisplayStatus, c: Color): void
+ notify_results(white_doubled_results: int, black_doubled_results: int): void
+ setup_board(in: std::istream&, is_eof_given: bool&): void
+ start_game(white: PlayerType, black: PlayerType): void
+ make_move(start: Coordinate, end: Coordinate, promotion: PromotionType): void
+ resign(): void
+ get_scores(): void
+ can_setup_board(): bool
+ can_start_game(): bool
+ can_make_move(): bool
+ can_resign(): bool
+ is_next_move_human(): bool
+ has_game(): bool
+ has_players(): bool
+ get_last_game(): std::shared_ptr<Game>
+ get_last_game_players(): std::shared_ptr<Players>
(0 ... *)
Game
Public/Protected/Design Pattern Attributes
Public Operations
+ init(default_board: bool = true): void
+ setup_board(in: std::istream&, is_eof_given: bool&): void
+ make_move(start: Coordinate, end: Coordinate, promotion: PromotionType): void
+ resign(): void
(1)
(1)
(1)
Board
Public/Protected/Design Pattern Attributes
+ piece_weights: std::map<PieceType, int>
Public Operations
+ create_piece(color: Color, square: Coordinate, type: PieceType): std::shared_ptr<Piece>
+ place_piece(color: Color, square: Coordinate, type: PieceType): void
+ remove_piece(square: Coordinate): void
+ verify_setup(): void
+ setup_default_board(): void
+ clean_board(): void
+ is_valid_move(start: Coordinate, end: Coordinate, Color C): MoveFlags
+ checkmate_valid_move(Coordinate start, Coordinate end, Color c): MoveFlags
+ make_move(start: Coordinate, end: Coordinate, promotion: PromotionType): std::string
+ add_piece(p: std::shared_ptr<Piece>): void
+ destroy_piece(p: std::shared_ptr<Piece>): void
+ is_promotion(start: Coordinate, end: Coordinate): bool
+ get_threatened_squares_by_color(s: std::set<Coordinate>&, c: Color): void
+ get_enpassant_coordinate(): Coordinate
+ is_valid_enpassant(start: Coordinate, end: Coordinate): bool
+ is_valid_castle(start: Coordinate, end: Coordinate): bool
+ is_double_pawn_move(start: Coordinate, end: Coordinate): bool
+ is_double_king_move(start: Coordinate, end: Coordinate): bool
+ add_enpassant_target(c: std::shared_ptr<Coordinate>): void
+ get_enpassant_taken_piece_coordinate(): Coordinate
+ get_enpassant_square_coordinate(c: Coordinate): std::shared_ptr<Coordinate>
+ get_castle_rook(c: Color, cs: CastleSide): Coordinate
+ get_all_valid_moves(s: std::set<std::pair<Coordinate, Coordinate>>&, c: Color): void
+ checkmate_get_all_valid_moves(std::set<std::pair<Coordinate, Coordinate>>& s, Color c): void
+ get_pieces(c: Color): std::set<std::shared_ptr<Piece>>
+ is_check(c: Color): bool
+ is_stalemate(c: Color): bool
+ is_checkmate(c: Color): bool
(1)
(0 ... 64)
(1)
Coordinate
Attributes
+ row: int
+ column: char
Operations
+ operator<(other: const Coordinate&) const: bool
+ operator==(other: const Coordinate&) const: bool
(1)
CastleSide
• "KINGSIDE"
• "QUEENSIDE"
PromotionType
• "KNIGHT"
• "BISHOP"
• "ROOK"
• "RESIGN"
• "QUEEN"
DisplayStatus
• "CHECKMATE"
• "STALEMATE"
• "CHECK"
• "RESIGN"
• "NONE"
Result
• "WHITE_WIN"
• "BLACK_WIN"
• "DRAW"
PlayerType
• "HUMAN"
• "LEVEL1"
• "LEVEL2"
• "LEVEL3"
• "LEVEL4"
• "LEVEL5"
Color
• "WHITE"
• "BLACK"
DisplayBoard
Public/Protected/Design Pattern Attributes
chess: std::weak_ptr<Chess>
Public Operations
+ virtual show() = 0: void
+ virtual show_status(s: DisplayStatus, c: Color) = 0: void
+ virtual show_results(white_doubled_results: int, black_doubled_results: int) = 0: void
(0 ... *)
(2)
Players
Attributes
+ white: std::shared_ptr<Player>
+ black: std::shared_ptr<Player>
(2)
PieceType
• "PAWN"
• "KNIGHT"
• "BISHOP"
• "ROOK"
• "QUEEN"
• "KING"
Piece
Public/Protected/Design Pattern Attributes
color: Color
location: Coordinate
type: PieceType
board: std::weak_ptr<Board>
directions: std::vector<std::pair<int, int>>
Public Operations
+ virtual get_valid_moves(s: std::set<Coordinate>&): void
+ virtual get_threatened_squares(s: std::set<Coordinate>&) = 0: void
+ is_valid_move(c: Coordinate): bool
+ single_move(directions: std::vector<std::pair<int, int>>, s: std::set<Coordinate>&): void
+ multiple_moves(directions: std::vector<std::pair<int, int>>, s: std::set<Coordinate>&): void
+ filter_moves(s: std::set<Coordinate>&): void
+ get_piece_character(): char
+ print_attack(): void
+ print_move(): void
(1)
(1)
GraphicsDisplay
Public/Protected/Design Pattern Attributes
Public Operations
+ show() override: void
+ show_status(s: DisplayStatus, c: Color) override: void
+ show_results(white_doubled_results: int, black_doubled_results: int) override: void
(1)
TextDisplay
Public/Protected/Design Pattern Attributes
Public Operations
+ show() override: void
+ show_status(s: DisplayStatus, c: Color) override: void
+ show_results(white_doubled_results: int, black_doubled_results: int) override: void
(1)
xwindow
Public/Protected/Design Pattern Attributes
+ rendered_board: std::vector<std::vector<char>>
Public Operations
+ fillGrid(x: int, y: int, colour: int = Black): void
+ drawPiece(x: int, y: int, piece: char): void
Player
Public/Protected/Design Pattern Attributes
game: std::shared_ptr<Game>
is_human: bool
color: Color
Public Operations
+ virtual make_move(start: Coordinate, end: Coordinate, promotion: PromotionType): void
+ virtual move(): void
+ resign(): void
Rook
Public/Protected/Design Pattern Attributes
Public Operations
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
Knight
Public/Protected/Design Pattern Attributes
Public Operations
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
Bishop
Public/Protected/Design Pattern Attributes
Public Operations
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
Queen
Public/Protected/Design Pattern Attributes
Public Operations
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
Pawn
Public/Protected/Design Pattern Attributes
Public Operations
+ get_valid_moves(s: std::set<Coordinate>&) override: void
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
+ get_captures(): std::vector<std::pair<int, int>>&
King
Public/Protected/Design Pattern Attributes
Public Operations
+ get_threatened_squares(s: std::set<Coordinate>&) override: void
+ in_check(): bool
+ in_checkmate(): bool
Computer
Public/Protected/Design Pattern Attributes
random_number(int lower, int upper, int size): int
Public Operations
+ move() override: void
+ execute_move(set<pair<Coordinate, Coordinate>> moves): void
+ execute_moves_based_on_priority(vector<set<pair<Coordinate, Coordinate>>> moves): void
Human
Public/Protected/Design Pattern Attributes
Public Operations
+ make_move(Coordinate start, Coordinate end, PromotionType promotion) override: void
LevelOne
Public/Protected/Design Pattern Attributes
Public Operations
+ move() override: void
LevelTwo
Public/Protected/Design Pattern Attributes
Public Operations
+ move() override: void
LevelThree
Public/Protected/Design Pattern Attributes
Public Operations
+ move() override: void
LevelFour
Public/Protected/Design Pattern Attributes
Public Operations
+ move() override: void
LevelFive
Public/Protected/Design Pattern Attributes
Public Operations
+ move() override: void
LEGEND
Abstract Class
Attributes
Operations
Concrete Class
Attributes
Operations
Struct
Attributes
Enumeration
• "VALUE_1"
• "VALUE_2"
• "VALUE_3"