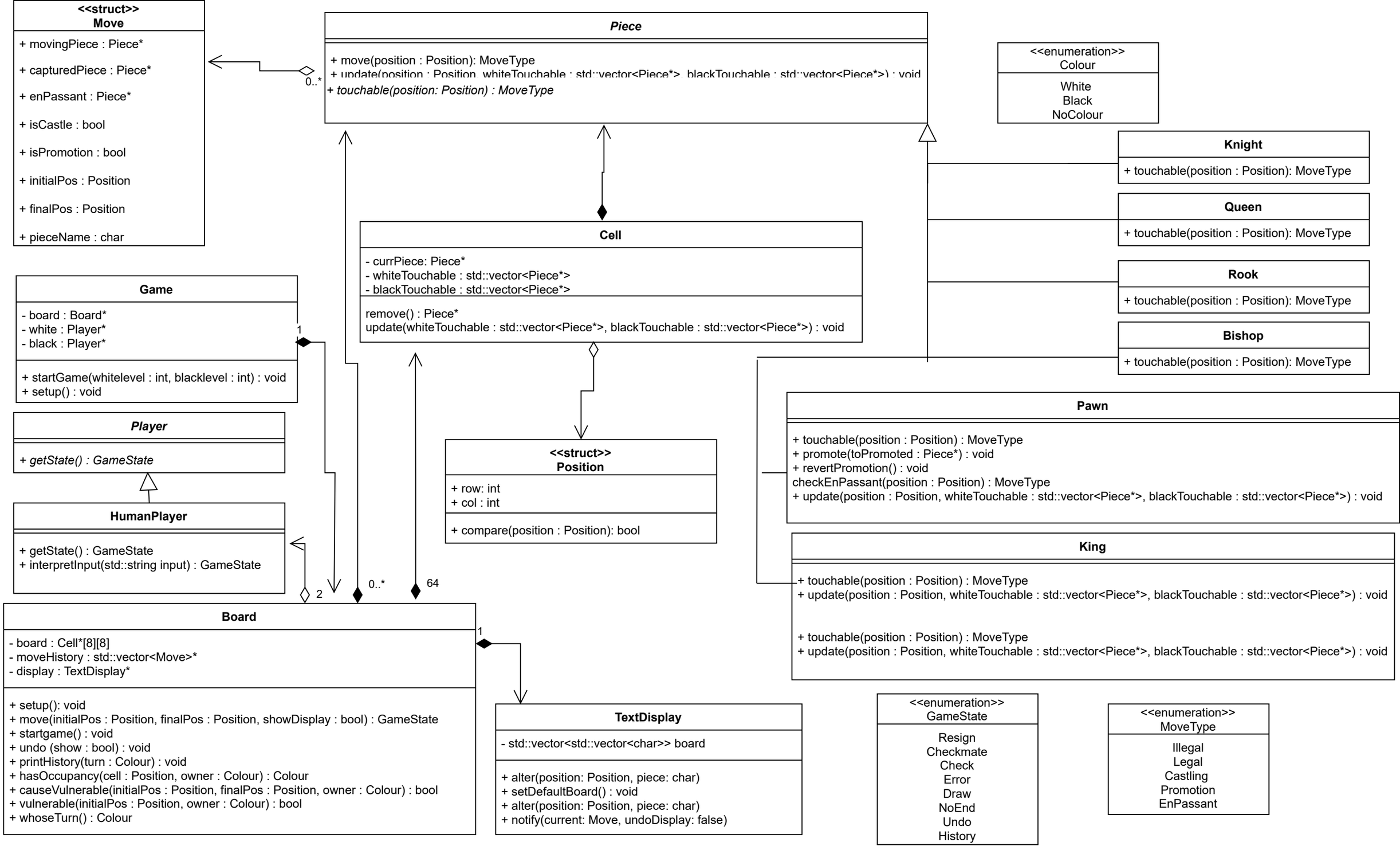
<<struct>>
Move
+ movingPiece : Piece*
+ capturedPiece : Piece*
+ enPassant : Piece*
+ isCastle : bool
+ isPromotion : bool
+ initialPos : Position
+ finalPos : Position
+ pieceName : char
0..*
Piece
+ move(position : Position): MoveType
+ update(position : Position, whiteTouchable : std::vector<Piece*>, blackTouchable : std::vector<Piece*>) : void
+ touchable(position: Position) : MoveType
<<enumeration>>
Colour
White
Black
NoColour
Knight
+ touchable(position : Position): MoveType
Queen
+ touchable(position : Position): MoveType
Rook
+ touchable(position : Position): MoveType
Bishop
+ touchable(position : Position): MoveType
Cell
- currPiece: Piece*
- whiteTouchable : std::vector<Piece*>
- blackTouchable : std::vector<Piece*>
remove() : Piece*
update(whiteTouchable : std::vector<Piece*>, blackTouchable : std::vector<Piece*>) : void
Game
- board : Board*
- white : Player*
- black : Player*
+ startGame(whitelevel : int, blacklevel : int) : void
+ setup() : void
1
Player
+ getState() : GameState
HumanPlayer
+ getState() : GameState
+ interpretInput(std::string input) : GameState
<<struct>>
Position
+ row: int
+ col : int
+ compare(position : Position): bool
Pawn
+ touchable(position : Position) : MoveType
+ promote(toPromoted : Piece*) : void
+ revertPromotion() : void
checkEnPassant(position : Position) : MoveType
+ update(position : Position, whiteTouchable : std::vector<Piece*>, blackTouchable : std::vector<Piece*>) : void
King
+ touchable(position : Position) : MoveType
+ update(position : Position, whiteTouchable : std::vector<Piece*>, blackTouchable : std::vector<Piece*>) : void
+ touchable(position : Position) : MoveType
+ update(position : Position, whiteTouchable : std::vector<Piece*>, blackTouchable : std::vector<Piece*>) : void
2
0..*
64
Board
- board : Cell*[8][8]
- moveHistory : std::vector<Move>*
- display : TextDisplay*
+ setup(): void
+ move(initialPos : Position, finalPos : Position, showDisplay : bool) : GameState
+ startgame() : void
+ undo (show : bool) : void
+ printHistory(turn : Colour) : void
+ hasOccupancy(cell : Position, owner : Colour) : Colour
+ causeVulnerable(initialPos : Position, finalPos : Position, owner : Colour) : bool
+ vulnerable(initialPos : Position, owner : Colour) : bool
+ whoseTurn() : Colour
1
TextDisplay
- std::vector<std::vector<char>> board
+ alter(position: Position, piece: char)
+ setDefaultBoard() : void
+ alter(position: Position, piece: char)
+ notify(current: Move, undoDisplay: false)
<<enumeration>>
GameState
Resign
Checkmate
Check
Error
Draw
NoEnd
Undo
History
<<enumeration>>
MoveType
Illegal
Legal
Castling
Promotion
EnPassant