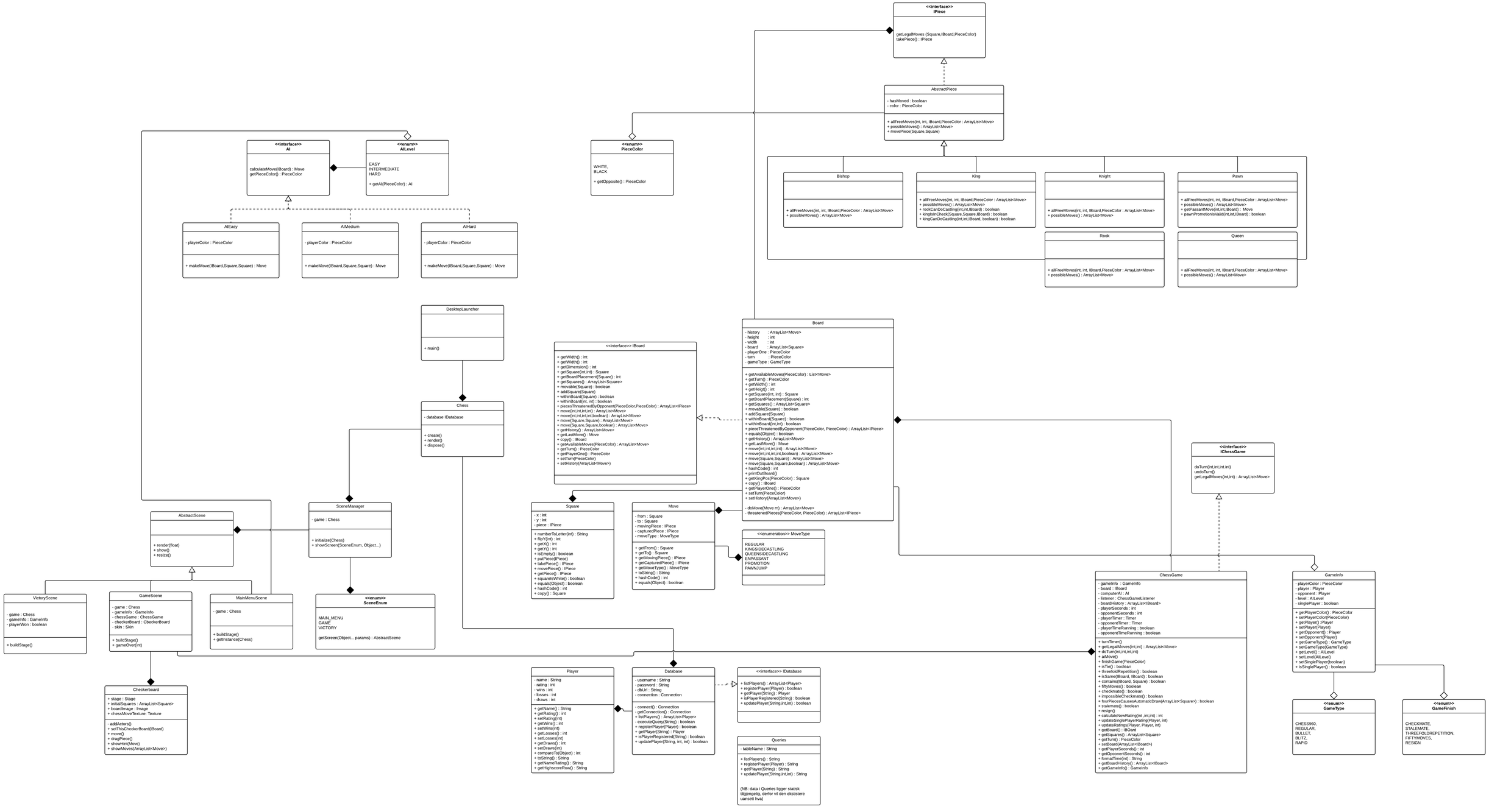
<<interface>>
IPiece
getLegalMoves (Square,IBoard,PieceColor)
takePiece() : IPiece
AbstractPiece
- hasMoved : boolean
- color : PieceColor
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
+ movePiece(Square,Square)
<<interface>>
AI
calculateMove(IBoard) : Move
getPieceColor() : PieceColor
<<enum>>
AILevel
EASY
INTERMEDIATE
HARD
+ getAI(PieceColor) : AI
<<enum>>
PieceColor
WHITE,
BLACK
+ getOpposite() : PieceColor
AIEasy
- playerColor : PieceColor
+ makeMove(IBoard,Square,Square) : Move
AIMedium
- playerColor : PieceColor
+ makeMove(IBoard,Square,Square) : Move
AIHard
- playerColor : PieceColor
+ makeMove(IBoard,Square,Square) : Move
Bishop
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
King
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
+ rookCanDoCastling(int,int,IBoard) : boolean
+ kingIsInCheck(Square,Square,IBoard) : boolean
+ kingCanDoCastling(int,int,IBoard, boolean) : boolean
Knight
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
Pawn
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
+ getPassantMove(int,int,IBoard) : Move
+ pawnPromotionIsValid(int,int,IBoard) : boolean
Rook
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
Queen
+ allFreeMoves(int, int, IBoard,PieceColor : ArrayList<Move>
+ possibleMoves() : ArrayList<Move>
DesktopLauncher
+ main()
Chess
- database IDatabase
+ create()
+ render()
+ dispose()
<<interface>> IBoard
+ getWidth() : int
+ getWidth() : int
+ getDimension() : int
+ getSquare(int,int) : Square
+ getBoardPlacement(Square) : int
+ getSquares() : ArrayList<Square>
+ movable(Square) : boolean
+ addSquare(Square)
+ withinBoard(Square) : boolean
+ withinBoard(int, int) : boolean
+ piecesThreatenedByOpponent(PieceColor,PieceColor) : ArrayList<IPiece>
+ move(int,int,int,int) : ArrayList<Move>
+ move(int,int,int,int,boolean) : ArrayList<Move>
+ move(Square,Square) : ArrayList<Move>
+ move(Square,Square,boolean) : ArrayList<Move>
+ getHistory() : ArrayList<Move>
+ getLastMove() : Move
+ copy() : IBoard
+ getAvailableMoves(PieceColor) : ArrayList<Move>
+ getTurn() : PieceColor
+ getPlayerOne() : PieceColor
+ setTurn(PieceColor)
+ setHistory(ArrayList<Move>)
Board
- history : ArrayList<Move>
- height : int
- width : int
- board : ArrayList<Square>
- playerOne : PieceColor
- turn : PieceColor
- gameType : GameType
+ getAvailableMoves(PieceColor) : List<Move>
+ getTurn() : PieceColor
+ getWidth() : int
+ getHeigt() : int
+ getSquare(int, int) : Square
+ getBoardPlacement(Square) : int
+ getSquares() : ArrayList<Square>
+ movable(Square) : boolean
+ addSquare(Square)
+ withinBoard(Square) : boolean
+ withinBoard(int,int) : boolean
+ pieceThreatenedByOpponent(PieceColor, PieceColor) : ArrayList<IPiece>
+ equals(Object) : boolean
+ getHistory() : ArrayList<Move>
+ getLastMove() : Move
+ move(int,int,int,int) : ArrayList<Move>
+ move(int,int,int,int,boolean) : ArrayList<Move>
+ move(Square,Square) : ArrayList<Move>
+ move(Square,Square,boolean) : ArrayList<Move>
+ hashCode() : int
+ printOutBoard()
+ getKingPos(PieceColor) : Square
+ copy() : IBoard
+ getPlayerOne() : PieceColor
+ setTurn(PieceColor)
+ setHistory(ArrayList<Move>)
- doMove(Move m) : ArrayList<Move>
- threatenedPieces(PieceColor, PieceColor) : ArrayList<IPiece>
<<interface>>
IChessGame
doTurn(int,int,int,int)
undoTurn()
getLegalMoves(int,int) : ArrayList<Move>
AbstractScene
+ render(float)
+ show()
+ resize()
SceneManager
- game : Chess
+ initialize(Chess)
+ showScreen(SceneEnum, Object...)
Square
- x : int
- y : int
- piece : IPiece
+ numberToLetter(int) : String
+ flipY(int) : int
+ getX() : int
+ getY() : int
+ isEmpty() : boolean
+ putPiece(IPiece)
+ takePiece() : IPiece
+ movePiece() : IPiece
+ getPiece() : IPiece
+ squareIsWhite() : boolean
+ equals(Object) : boolean
+ hashCode() : int
+ copy() : Square
Move
- from : Square
- to : Square
- movingPiece : IPiece
- capturedPiece : IPiece
- moveType : MoveType
+ getFrom() : Square
+ getTo() : Square
+ getMovingPiece() : IPiece
+ getCapturedPiece() : IPiece
+ getMoveType() : MoveType
+ toString() : String
+ hashCode() : int
+ equals(Object) : boolean
<<enumeration>> MoveType
REGULAR
KINGSIDECASTLING
QUEENSIDECASTLING
ENPASSANT
PROMOTION
PAWNJUMP
VictoryScene
- game : Chess
- gameInfo : GameInfo
- playerWon : boolean
+ buildStage()
GameScene
- game : Chess
- gameInfo : GameInfo
- chessGame : ChessGame
- checkerBoard : CbeckerBoard
- skin : Skin
+ buildStage()
+ gameOver(int)
MainMenuScene
- game : Chess
+ buildStage()
+ getInstance(Chess)
<<enum>>
SceneEnum
MAIN_MENU
GAME
VICTORY
getScreen(Object... params) : AbstractScene
ChessGame
- gameInfo : GameInfo
- board : IBoard
- computerAI : AI
- listener : ChessGameListener
- boardHistory : ArrayList<IBoard>
- playerSeconds : int
- opponentSeconds : int
- playerTimer : Timer
- opponentTimer : Timer
- playerTimeRunning : boolean
- opponentTimeRunning : boolean
+ turnTimer()
+ getLegalMoves(int,int) : ArrayList<Move>
+ doTurn(int,int,int,int)
+ aiMove()
+ finishGame(PieceColor)
+ isTie() : boolean
+ threefoldRepetition() : boolean
+ isSame(IBoard, IBoard) : boolean
+ contains(IBoard, Square) : boolean
+ fiftyMoves() : boolean
+ checkmate() : boolean
+ impossibleCheckmate() : boolean
+ fourPiecesCausesAutomaticDraw(ArrayList<Square>) : boolean
+ stalemate() : boolean
+ resign()
+ calculateNewRating(int ,int,int) : int
+ updateSinglePlayerRating(Player, int)
+ updateRatings(Player, Player, int)
+ getBoard() : IBOard
+ getSquares() : ArrayList<Square>
+ getTurn() : PieceColor
+ setBoard(ArrayList<IBoard>)
+ getPlayerSeconds() : int
+ getOpponentSeconds() : int
+ formatTime(int) : String
+ getBoardHistory() : ArrayList<IBoard>
+ getGameInfo() : GameInfo
GameInfo
- playerColor : PieceColor
- player : Player
- opponent : Player
- level : AILevel
- singlePlayer : boolean
+ getPlayerColor() : PieceColor
+ setPlayerColor(PieceColor)
+ getPlayer() :Player
+ setPlayer(Player)
+ getOpponent() : Player
+ setOpponent(Player)
+ getGameType() : GameType
+ setGameType(GameType)
+ getLevel() : AILevel
+ setLevel(AILevel)
+ setSinglePlayer(boolean)
+ isSinglePlayer() : boolean
<<enum>>
GameType
CHESS960,
REGULAR,
BULLET,
BLITZ,
RAPID
<<enum>>
GameFinish
CHECKMATE,
STALEMATE,
THREEFOLDREPETITION,
FIFTYMOVES,
RESIGN
Checkerboard
+ stage : Stage
+ initialSquares : ArrayList<Square>
+ boardImage : Image
+ chessMoveTexture: Texture
- addActors()
+ setThisCheckerBoard(IBoard)
+ move()
+ dragPiece()
+ showHint(Move)
+ showMoves(ArrayList<Move>)
Player
- name : String
- rating : int
- wins : int
- losses : int
- draws : int
+ getName() : String
+ getRating() : int
+ setRating(int)
+ getWins() : int
+ setWins(int)
+ getLosses() : int
+ setLosses(int)
+ getDraws() : int
+ setDraws(int)
+ compareTo(Object) : int
+ toString() : String
+ getNameRating() : String
+ getHighscoreRow() : String
Database
- username : String
- password : String
- dbUrl : String
- connection : Connection
- connect() : Connection
- getConnection() : Connection
+ listPlayers() : ArrayList<Player>
- executeQuery(String) : boolean
+ registerPlayer(Player) : boolean
+ getPlayer(String) : Player
+ isPlayerRegistered(String) : boolean
+ updatePlayer(String, int, int) : boolean
<<interface>> IDatabase
+ listPlayers() : ArrayList<Player>
+ registerPlayer(Player) : boolean
+ getPlayer(String) : Player
+ isPlayerRegistered(String) : boolean
+ updatePlayer(String,int,int) : boolean
Queries
- tableName : String
+ listPlayers() : String
+ registerPlayer(Player) : String
+ getPlayer(String) : String
+ updatePlayer(String,int,int) : String
(NB: data i Queries ligger statisk
tilgjengelig, derfor vil den ekststiere
uansett hva)