BubbleChess Client Side

# GUI (com.bubblechess.gui)

**GameBoard**

* Board: BoardSquare[][]
* Refresh(): void
* GetSquare(x,y): BoardSquare
* SetSquare(x,y,BoardSquare): void
* Should we add functionality to pull board here?

**BoardSquare (Interface)**

* GetPieceImage(): Image? – Do we draw this in the implemented object?

**ChessBoardSquare (Implements BoardSquare)**

* Enum PIECETYPES {PAWN, BISHOP, etc.}
* Type: PIECETYPE
* Color: “black” or “white”?
* Highlighted: Boolean
* GetPieceImage(): Image?

# Client Side (com.bubblechess.client)

**Game**

* GameID: Int
* Player1: User
* Player2: User
* Board: GameBoard
* MoveHistory: Moves[]

**Board (interface)**

* GetBoardWidth(): int
* GetBoardHeight(): int
* ApplyMove(Move): Boolean
* ApplyMoveCloning(Move): Board
* GetAllMoves(): Move[]
* GetAllMoves(Player: int): Move[]
* GetMovesForPeice(x,y): Move[]

**ChessBoard (Implements Board)**

* Board: ChessPiece[][] (8x8)
* Captured: ChessPeice[]
* State: State ENUM (White move, black move, endgame)

**ChessPiece – We need to figure out the details for how to determine moves. How do we represent possible directions? I don’t think we want the piece to have knowledge of the board**

* Name: String
* Create(): void
* Destroy(): ChessPiece
* Capture(): void
* MoveIsPossible(int[] from, int[] to): boolean

**Move**

* MoveID: Int
* MoveNumber: Int
* CoordinateFrom: int[]
* CoordinateTo: int[]
* Player: int

Design decisions made during implementation:

* Board object will know the state, Game will use board’s state to verify if moves are possible, etc.
* A utility class will be created to push the moves to the server
* Need to figure out how to choose game ID and user flow
* Our game extensibility will be limited to two-person board games

Work for Eric:

* Implement each chessPiece
* Add pieces to board in init()
* Catch exceptions in validMove() by using instanceof()
* Apply logic for special cases
* Handle state changes and populate states