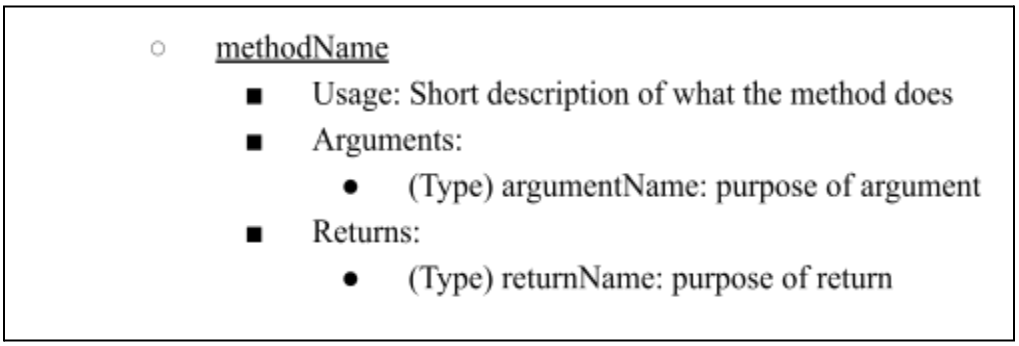
# Section 0: Document Guide

## Project

- https://github.com/JonathanUhler/Java-Chess

## Conventions Used Throughout

- **Classes**: bold
  - Follow the format: name, explanation
- *Constructors*: italics
  - Follow the format: name, explanation, arguments (if applicable)
- Methods: underline
  - Follow the format: name, explanation, arguments (if applicable), returns (if applicable)
  - 
    
- `Excerpts from code`: Courier New font

## Sections

# Section 1: Chess.java

- **Chess: Main class of the project**
  - main
    - Usage: The main method (entry point for the program. Initializes the board, settings, and starting position.
    - Arguments:
      - None of note
    - Returns:
      - None

# Section 2: Board.java

- **Board: Keeps track of the board and pieces on the board. Handles moving pieces and the boardstate**
  - getPieceTracker
    - Usage: Returns a piece tracker based on a color and type of piece
    - Arguments:
      - (int) pieceType: the type of piece
      - (int) pieceColor: the color of the piece)
    - Returns:
      - The correct piece tracker
  - loadPosition
    - Usage: Loads a given position into PieceTrackers that will be used to place the pieces onto the board
    - Arguments:
      - (String) fen: the FEN string to load
    - Returns:
      - None
  - initBoard
    - Usage: Initializes information about the state of the board and the pieces (including PieceTrackers)
    - Arguments:
      - None
    - Returns:
      - None
  - makeMove
    - Usage: Makes a move on the board
    - Arguments:
      - (Move) move: the move to be made
      - (boolean) isGhost: should the move be played on the board, or is this move just for check(mate) search
    - Returns:
      - None
  - changePlayer
    - Usage: Changes information about which player is currently taking their turn
    - Arguments:
      - None
    - Returns:
      - None

- checkState
  - Usage: Checks the state of the game to determine how many legal moves the current player has and if they are in check
  - Arguments:
    - None
  - Returns:
    - None
- clone
  - Usage: Creates a deep copy of the current Board object into a new Board object
  - Arguments:
    - None
  - Returns:
    - (Board) newBoard: the deep copy

# Section 3: BoardManager.java

- **BoardManager: Contains useful methods for the board**
  - playSound
    - Usage: Plays a sound given a sound file
    - Arguments:
      - (String) soundFile: the sound file path
    - Returns:
      - None
  - frameRelativeMousePosition
    - Usage: Takes in the position of a JFrame and the mouse position and returns the mous position relative to the JFrame instead of relative to the entire computer screen
    - Arguments:
      - (JFrame) frame: the JFrame the position should be based on
      - (Point) screenRelativeMousePos: the mouse position relative to the entire screen
    - Returns:
      - (Point) frameRelativeMousePos: the mouse position relative to the JFrame frame

# Section 4: FenUtility.java

- **FenInfo: Information about the loaded FEN string**
  - *FenInfo*
    - Usage: Sets the number of tiles to 64
    - Arguments:
      - None
- **FenUtility: Manges FEN strings**
  - loadPositionFromFen
    - Usage: Loads a given board position from a FEN string
    - Arguments:
      - (String) fen: the fen string to load
    - Returns:
      - (FenInfo) fenInfo: the information about the current fen position
  - buildFenFromPosition
    - Usage: Takes information about the current board position and returns the position's FEN string
    - Arguments:
      - None
    - Returns:
      - (String) fen: the fen string for the current position
  - changePlayerPerspective
    - Usage: Properly reverses a fen string to show the perspective of the game from the other player
    - Arguments:
      - (String) fen: the fen to reverse
    - Returns:
      - (String) fenReversed: the reversed fen

# Section 5: GameStateUtility.java

- **GameStateUtility: Handles wins, draws, and other details of figuring out the current game-state**
  - actOnGameState
    - Usage: Acts on the state of the game
    - Arguments:
      - (int) numLegalMoves: the number of legal moves the player has
      - (boolean) inCheck: whether or not the player is in check
    - Returns:
      - The state of the game represented by an integer
  - gameOver
    - Usage: Ends the game with a popup
    - Arguments:
      - (String) endState: a message to describe how the game ended
    - Returns:
      - None
  - addEndText
    - Usage: Creates a JLabel to explain how the game ended
    - Arguments:
      - (String) text: text to display in the label
    - Returns:
      - JLabel with the specified text
  - addCloseButton
    - Usage: Creates a JButton to close the popup
    - Arguments:
      - None
    - Returns:
      - (JButton) closeButton: the JButton to close the popup
  - createEndWindow
    - Usage: Create the JFrame
    - Arguments:
      - None
    - Returns:
      - None

# Section 6: Graphics.java

- **Graphics: Draws graphical elements onto the screen**
  - drawPosition
    - Usage: Draws the pieces onto the board
    - Arguments:
      - None
    - Returns:
      - None
  - drawBoard
    - Usage: Draws the board
    - Arguments:
      - (ArrayList<Integer>) highlightTiles: a list of tiles to highlight (intended for highlighting legal moves)
    - Returns:
      - None
  - createApplication
    - Usage: Creates the JFrame
    - Arguments:
      - None
    - Returns:
      - None

# Section 7: JSONUtility.java

- **JSONUtility: A custom JSON handler**
  - stringToDictionary
    - Usage: The primary method of the JSON class. Converts a string to a dictionary
    - Arguments:
      - (String) JSONString: the string to convert
    - Returns:
      - (Hashmap<String, String>) JSONDict: the converted dictionary
  - write
    - Usage: Writes data to a file
    - Arguments:
      - (String) file: the directory to write to
      - (String) data: the data to write to the specified file
    - Returns:
      - None
  - read
    - Usage: Reads a file and returns its contents as a string array
    - Arguments:
      - (String) file: the file to read
    - Returns:
      - (List<String>) data: the data as a string from the file
  - removeCharAt
    - Usage: Removes a character from a certain position in a string
    - Arguments:
      - (String) str: the string to edit
      - (int) n: the index of the character to be removed
    - Returns:
      - The new string

# Section 8: Move.java

- **MoveFlag: Different flags a move can have**
- **Move: Contains information about a move on the board**
  - *Move*
    - Usage: Constructor 1 for Move class
    - Arguments:
      - (int) moveValue: the value of the move (16-bit binary integer that holds the starting tile, ending tile, and flag)
  - *Move*
    - Usage: Constructor 2 for the Move class
    - Arguments:
      - (int) startTile: the starting tile for the moving piece
      - (int) endTile: the desired tile for the moving piece
  - *Move*
    - Usage: Constructor 3 for the Move class
    - Arguments:
      - (int) startTile: the starting tile for the moving piece
      - (int) endTile: the desired tile for the moving piece
      - (int) flag: special flag for the move
  - startTile
    - Usage: Returns the start tile
    - Arguments:
      - None
    - Returns:
      - The start tile
  - endTile
    - Usage: Returns the end tile
    - Arguments:
      - None
    - Returns:
      - The end tile
  - isPromotion
    - Usage: Whether or not the move is a pawn promotion
    - Arguments:
      - None
    - Returns:
      - Whether or not the move is a pawn promotion
  - isCastle
    - Usage: Whether or not the move is a castle

        - Arguments:
            - None
        - Returns:
            - Whether or not the move is a castle
    - moveFlag
        - Usage: Returns the move flag
        - Arguments:
            - None
        - Returns:
            - The move flag

# Section 9: MoveData.java

- **MoveData: Initializes some precomputer move data**
  - *MoveData*
    - Usage: Computes some basic information about moves
    - Arguments:
      - None

# Section 10: MoveUtility.java

- **MoveUtility: Handles move generation**
  - returnMoveValues
    - Usage: Returns a list of move values from a list of move objects
    - Arguments:
      - (List<Move>) listOfMoves: the list of move objects
    - Returns:
      - (List<Integer>) moveVals: the list of move values
  - returnEndingTiles
    - Usage: Returns a list of ending tiles from a list of move objects
    - Arguments:
      - (List<Move>) listOfMoves: the list of move objects
    - Returns:
      - (List<Integer>) endTiles: the list of end tiles
  - generateMoves
    - Usage: Generates legal moves
    - Arguments:
      - (Board) board: the board to generate moves for
    - Returns:
      - (List<Move>) psuedoLegalMoves: a list of legal moves
  - generateSlidingMoves
    - Usage: Generates the legal moves for sliding pieces (bishops, rooks, and queens)
    - Arguments:
      - (int) startTile: the starting tile for the piece being checked
      - (int) piece: the sliding piece being checked
    - Returns:
      - (List<Move>) movesGenerated: a list of sliding moves generated
  - generateKingMoves
    - Usage: Generates legal moves for kings
    - Arguments:
      - (int) startTile: the starting tile for the piece being checked
    - Returns:
      - (List<Move>) movesGenerated: a list of king moves generated
  - generateKnightMoves
    - Usage: Generates legal moves for knights
    - Arguments:
      - (int) startTile: the starting tile for the piece being checked
    - Returns:

            - (List<Move>) movesGenerated: a list of knight oves generated
    - generatePawnMoves
        - Usage: Generates legal moves for pawns
        - Arguments:
            - (int) startTile: the starting tile for the piece being checked
        - Returns:
            - (List<Move>) movesGenerated: a list of pawn moves generated
    - generatePawnCaptures
        - Usage: Generates legal captures for pawns
        - Arguments:
            - (int) startTile: the starting tile for the piece being checked
        - Returns:
            - (List<Move>) movesGenerated: a list of pawn captures generated
    - generateEnPassantCaptures
        - Usage: Generates legal en passant captures for pawns
        - Arguments:
            - (int) startTile: the starting tile for the piece being checked
        - Returns:
            - (List<Move>) movesGenerated: a list of en passant captures generated

# Section 11: Piece.java

- **Piece: Framework to create instances of each piece**
  - checkColor
    - Usage: Find the color of a piece using bitwise AND
    - Arguments:
      - (int) piece: the piece to find the coor of
      - (int) color: this is usually Piece.White. The purpose of this function is to compare the color against colorAnd and return the result
      - (boolean) colorOnlyShiften: whether or not the color should be shiften to return a value of (0 or 1) or (8 or 16)
    - Returns:
      - The comparison of the color of the piece
  - pieceColor
    - Usage: Finds the color of a piece
    - Arguments:
      - (int) piece: the piece to find the color of
    - Returns:
      - The color of the piece
  - pieceType
    - Usage: Finds the type of a piece
    - Arguments:
      - (int) piece: the piece to find the type of
    - Returns:
      - The type of the piece
  - checkSliding
    - Usage: Figures out if a piece is a sliding piece or not
    - Arguments:
      - (int) piece: the piece to find the behavior of
    - Returns:
      - Whether or not the piece is sliding

# Section 12: PieceTracker.java

- **PieceTracker: Keeps track of all pieces (adding pieces to the board, removing pieces, changing piece location)**
  - *PieceTracker*
    - Usage: Facilitates the creation of a new PieceTracker for a given type of piece
    - Arguments:
      - (int) maxCountPerPieceType: the maximum amount of a given type of piece possible in a game
      - (int) color: the color of the piece tracker
      - (int) type: the type of the piece tracker
  - addPieceToTile
    - Usage: Adds a new piece to a given tile
    - Arguments:
      - (int) tile: the index of the tile to add a piece to
    - Returns:
      - None
  - removePieceFromTile
    - Usage: Removes an existing piece from a given tile
    - Arguments:
      - (int) tile: the index of the tile to remove a piece from
    - Returns:
      - None
  - movePiece
    - Usage: Moves a piece from a starting tile to an ending tile
    - Arguments:
      - (int) startingTile: the tile index the piece is currently on
      - (int) endingTile: the tile index the piece should move to
    - Returns:
      - None
  - clone
    - Usage: Clones hte current PieceTracker object to a new PieceTracker object
    - Arguments:
      - None
    - Returns:
      - (PieceTracker) newPieceTracker: cloned piece tracker

# Section 13: PromotionUtility.java

- **PromotionUtility: Handles details of promotions**
  - getPromotionPiece
    - Usage: Returns the piece chosen for promotion
    - Arguments:
      - None
    - Returns:
      - (int) promoteTo: the piece to promote to
  - addPromotionmenu
    - Usage: Creates a JComboBox to select the piece to promote to
    - Arguments:
      - None
    - Returns:
      - (JComboBox) promotionType: the JComboBox
  - createPromotionWindow
    - Usage: Creates the JFrame
    - Arguments:
      - None
    - Returns:
      - None

# Section 14: Settings.java

- **Settings: Displays settings onto the screen**
  - drawSettings
    - Usage: Compiles the different settings and options and returns them as a JPanel
    - Arguments:
      - None
    - Returns:
      - (JPanel) settingsPanel: a JPanel containing the settings and options for the game
  - playerColor
    - Usage: Creates the combo box for the players desired starting color
    - Arguments:
      - None
    - Returns:
      - (JComboBox<String>) playerColor: the combo box for the player color choices
  - boardTheme
    - Usage: Creates the combo box for the board theme
    - Arguments:
      - None
    - Returns:
      - (JComboBox<String>): boardThemesDropdown: the JComboBox for the board theme
  - customFenPosition
    - Usage: Creates a text field for a custom fen position
    - Arguments:
      - None
    - Returns:
      - (JTextField) customFenBox: the JTextField for the custom fen position
  - changePerspective
    - Usage: Changes the perspective of the game (which player is being viewed). Note, the perspective also changes after each move automatically
    - Arguments:
      - None
    - Returns:
      - (JButton) changePerspective: the button to change perspective

- showLegalMoves
  - Usage: Toggles whether or not legal moves are shown
  - Arguments:
    - None
  - Returns:
    - (JButton) toggleLegalMoves: the button that toggles legal moves
- newGame
  - Usage: Starts a new game
  - Arguments:
    - None
  - Returns:
    - (JButton) newGame: the button that toggles legal moves
- pieceMaterial
  - Usage: Shows piece advantage/score
  - Arguments:
    - None
  - Returns:
    - (JLabel) material: a label with the piece materials