Brian Lasker – CS4330 – 2/12/18

Checker Board Data Model Concept

<u>Fields</u>

- firstPlayerName(String:holds first players name)
- secondPlayerName(String: holds second players name)
- firstPlayerColor(Color = red)
- secondPlayerColor(Color = black)
- boardWidth(double: holds the width of the board for methods of resizing and gameboard creation)
- boardHeight(double: same purpose as boardWidth but holds the height instead)
- firstPlayerPieces(int: holds the number of pieces for the first player, when 0 that player loses, starts at 12)
- secondPlayerPieces(int: same as firstPlayerPieces)
- piecesPositions (2d array for every checker on the board and holds “pieces” in the location in the 2d array corresponding to where they are on the board. Also holds characteristics of color and if it is a king or not)
- pieceWidth and pieceHeight (double: dimensions of the piece)
- rectWidth and rectHeight (double: dimensions of each available space)
- anchorPane (AnchorPane: holds the board)
- numRows and numCols (final int: holds the number of rows and cols for the board)

<u>Methods</u>

- Constructor (sets the variables of the CheckerBoard class, takes in numRows, numCols, boardWidth, and boardHeight)
- Build() (returns anchorPane which is the board with the rectangles added and pieces in each position)
- isKing(row, col) (returns Boolean. If piece at the row and col is holds the value of king then it returns true otherwise returns false)
- validMove(row, col, color)(returns Boolean if move is valid. Move is determined valid if the space is open. The validity for the dark player it is row + 1 and col + 1 or row – 1 and col + 1 and for the light player it is row + 1 col – 1 and row – 1 and col – 1. This works because it allows each normal piece to move in diagonals. Also error trapped so the piece can not move outside of the array by comparing it to numRows and numCols for the designated new position. If the piece is a king though any of the 4 diagonals are valid and would be determined if is isKing() returns true)
- move() (returns void. Calls validMove to check if the space is a viable option than it moves it and changes the position fields for that piece. Also checks to see if a piece is jumped or if the move will cause it to become a king. Also checks all surrounding moves and if no valid move can be made for any of the pieces left than breaks out of the function and notifies a stalemate)

- isJump(row, col, color) (returns Boolean. If a piece is obstructing the move a piece and it is the opposite color then it bumps the row and col accordingly to the viable spot and returns true otherwise returns fales. If true it will also remove that piece from the array and lower the count of the corresponding players piece count)
- isVictory()(returns Boolean. Gets called after every move and if one the players piece variable is = 0 then the other player is declared a winner and a notification shows up)
- kingMe(row, col , color) (returns boolean. If the row and col match up with the other side of the board than that piece gains king status and returns true)
- firstMove() (returns int. does a random number generation if it returns 1 than firstPlayer is selected and if it is a 2 than the secondPlayer goes)
- Standard Getters and Setters for the fields in the class so that they can be accessed by the controller for code simplicity and functionality