CheckerBoard Data Model Concept - David Dean

-Objects and Methods

- Player
  - Properties
    - Int wins/ losses
    - Color
    - String name
    - Bool isTurn
    - Int numPiecesRemaining
    - Bool cantMove
  - Methods
    - getUserName/ setUserName
    - getColor
    - addWin/ addLoss
    - saveBoard
- Board
  - Properties
    - 2D array.
      - Game state will be stored in a text file, in order to save and load games that aren’t finished
    - Int numRows/ numCols
  - Methods
    - setupBoard
    - updateBoard
    - saveBoard
    - setActivePiece
    - getLegalMoves
- GamePiece
  - Properties
    - Int index
    - Bool isKing
    - Bool isSelected
      - Player selects a piece, and then the space(s) they can move the piece to are highlighted
    - Int xIndex/ yIndex
    - Bool isPlayable
  - Methods
    - movePiece
    - removePiece

-Outline

- I am going to implement the Checkers game using coordinated for each square in the Board. Ranging from (0,0) all the way up to (7,7) for the standard board (range will obviously adjust depending on the Grid layout the user chooses). The game board will ultimately be a 2D array Upon opening the game, the first thing that the user will have to decide upon opening the application is the players (PvP, PvC, CvC etc). They can decide if player one is going to be a person or computer, along with player 2, although they must decide this before the game will actually be played. If the user decides to make the player(s) human, they can also assign names, otherwise they will simply be called Player 1 and Player 2. The top player (Player 1) will be assigned black/ dark pieces

-Determine possible moves/ illegality

- A piece will either have 0, 1, or 2 legal moves at any point in time (assuming they isKing = false). If the player is black (starting at the top), the only legal moves will be the (xIndex +/- 1, yIndex+1) → 2 at most. Upon calling getLegalMoves, the function will also check if there is already a game piece in the one or two potential places to go; and if a space is occupied, the user will not be able to move to said piece. If a piece becomes a king, it will have the ability to make the moves of a dark and light piece (down and up)

-Determine winner

- Whenever a player is killed, their numRemainingPieces will decrement by one, and once it reaches 0, the other player will be deemed winner. If a player has no possible remaining moves and it's their turn, the game will stop: the person with more pieces left will be deemed winner (this is handling the draw).

-UI

- I will be implemented the same UI posted on canvas; more specifically the ones with coordinated, like I mentioned above