Data Model

- What data will your system deal with to meet the user’s needs? Define this in terms of **data only** - (classes, instance variables, enums) - **no logic yet**
    - Classes:
        - Game
            - Variables
                - Chessboard
                - File
                - List of moves
            - Methods
                - openPGN(filePath)
                - manageGraphics()
                - drawMove()
                - Move calls () chessboard.move(..)
        - Abstract Class ChessPiece
            - Variables
                - Abstract Boolean color
                - Abstract Image
            - Methods
                - Abstract Boolean isLegal (int x1, int y1, int x2, int y2)
                -
        - Classes extending ChessPiece, one for each chess piece
            - Variables
                - Boolean color
                - Image
            - Methods
                - consturctors(boolean isWhite)
                - Boolean isLegal (int x1, int y1, int x2, int y2)
                - getters
                - King, rook: boolean isMoved
        - Tree
            - Variables
                - root
                - Element
            - Methods
                - addMove(move)
                - addAlternateMove(move)
                - addChild(element)
                - getParent()
                - getChildren()
                - getMove()
        - Chessboard

        - Variables
            - Tree of moves to the chessboard
            - ChessPiece[8][8] board
        - Methods
            - Move(int x1, int y1, ChessPiece, int x2, int y2)
            - isLegaMovel(int x1, int y1, ChessPiece, int x2, int y2, boolean isWhiteTurn)
            - isPossibleMovel(int x1, int y1, ChessPiece, int x2, int y2, boolean isWhiteTurn)
            - isCheck(int x1, int y1, ChessPiece, int x2, int y2)
            - initializeBoard()
            - List<Piece> getPiecesAt(int x1, x2)
            - List<ChessPiece> getPiecesOfColor(boolean isWhiteTurn)
            - void MakeNewLine
            - void deleteLine
            - boolean isClear(int x1, int y1, int x2, int y2)
            - void createMove (int x1, int y1, int x2, int y2, boolean isWhiteTurn)
            - List<ChessPiece> getEnemyPieces(ChessPiece[][] board, boolean isWhiteTurn)
            - isCapture(int x1, int y1, int x2, int y2)
    - Move (command)
        - Variables
            - Txt - moves and comments
            - lambda function to do and undo the moves onto the chessboard
        - Methods
            - Undo a move
            - Annotate
            - Do a move

- **What data structures should you use to store and access your data?** Decide based on how the user will use the system; pick the data structures that **work best for what the user wants to accomplish**
    - We need a tree to model the moves

Implementation

Step 1: Create skeleton classes
- To model your data
- Method signatures for functionality

Step 2: Test Driven Development – Tests before application logic!

Step 3: Application logic

Step 4..N: Iterate, Iterate, Iterate…