Model

8x8 Array of type Gamepiece. Stores null for empty spaces (Or perhaps use a phantom piece? Undecided)

Gamepiece

- canMoveTo - return list of positions to which the piece can move
- moveTo - if the piece canMoveTo a given square, move there. Take enemy piece if there is one.
- Stores position x and position y (either as 2 ints or a Point2D, to be decided)
- Stores colour
- Boolean taken - whether or not the piece has been captured, and therefore removed from the board (to be used for the list of taken pieces on the side of the board)
- Boolean hasMoved - Tracks whether or not a piece has moved. Used for Pawns, Bishops, and Kings (Double move for Pawns, and castling for Bishop/King)
- Boolean isThreatening - whether or not the piece canMoveTo the opposite King. Run after move on all pieces to see if there is check present.

Turn

- Switch after a move is made. Possible implementation: if currentPlayer = White, currentPlayer = Black.
- Turn counter - potential feature, not necessary.

Clock

- Potential feature, not necessary.
- Each player starts with X amount of time, then gets Y additional time when their turn starts. Perhaps have a dialog on game launch to set X and Y values.
- Lose if time runs out on your turn.

Player Class

- Stores colour (This will be the name of the instance of the class, probably)
- Stores inCheck
- Stores opposite (Color that is not this piece, possibly hardcode)

View

8 x 8 array of Buttons.

- Checkerboard pattern
- List of taken pieces for each colour on the side.
- Resizable window
- Display A-H, 1-8 on the axes
- Buttons for concede, agree to draw, concede

Update

- Refresh the board
- Add taken pieces to the list on the side (if implemented)

Application

Game loop, event handlers

Game Loop

- At the start of the turn, check for loss (King cannot move and is threatened), stalemate (King cannot move but is not threatened, or the same board state has been repeated three times (Not sure if we're going to implement this)), or check.
- If player is in check, set inCheck = true.
- If inCheck is true, check all currentPlayer's pieces for canMoveTo something that ends check (take or block threatening piece). Disable selection of other pieces. Else enable all pieces
- currentPlayer makes a move, end turn, switch players.

Eventhandler (Board)

- if this space has a piece on it, select it. Else do nothing
- Highlight the selected piece, and possible move locations (as per the function canMoveTo). If player is in check, only moves that end check are legal, disable others.
- If the selected piece is clicked again, deselect it. If a legal move location is clicked, moveTo that location. Else do nothing

Eventhandler (Quit)

- Prompt "Are you sure?" If yes, close the program. If no, close the prompt.

Eventhandler (Agree to a Draw)

- Prompt "Does <currentPlayer.colour> wish to draw?" If yes, prompt 'Does <currentPlayer.opposite wish to draw?" If yes, display Draw message, and end the game.

Eventhandler (Concede)

- Prompt "Are you sure?" If yes, currentplayer.opposite wins. Display victory message, and end the game.