Questions:

How to grab data from the board. When a checker is clicked, how do we get its position and index?

Essential Functions:

Collect and Display Player Names:

-Upon App Start, Provide Input Fields to Collect Player Names and a Start Button to trigger functions to collect the info.

Determine who moves first and gets assigned the black (dark) pieces:

-To determine which player goes first, a simple rand() function will be used. The user will be presented with a coin flip graphic.

Display a board with pieces the user can interact with if one or both players are human.

-The initial board is created on start.

Data Model

The board must be equal in rows and columns. A single Array will contains each Square on the board. Each Square will be represented as an Object. Each Square Object will container:

Square Variables:

* Color squareColor
* Checker checker
* Int index;

Square Functions:

* getSquareColor
* getIndex
* getChecker
* setChecker

Checker Variables:

* Color checkerColor
* CheckerType checkerType (Normal or King)
* CheckerSide checkerSide (Top or Bottom)
* int currentRow

Checker Functions

* getColor
* getRow
* getCheckerType
* setCheckerType

It would be ideal for the player to have their piece to start at the bottom of the board, regardless of color. To maintain our algorithm, we will

Note, “forward” will be represented as forwardRight and forwardLeft.

**Illegal Moves**

* Blocked: A Checker cannot move into a space that already has a checker.
* OffBoard: A Checker cannot move off of the board or into red squares.
* FriendlyFire: A Checker cannot jump a checker of the same color.
* Direction: A checker cannot move backwards if it is not kinged.

**Legal Moves:**

* Forwards: A Checker may move forward into an empty square.
* Backwards: If a checker is kinged, the checker may move backwards.
* Jump: A checker may jump over a checker of a different color if the enemy checker is 1 row forward (or backward if kinged) and the space beyond the enemy checker is opened. Note a checker *must* jump multiple times as long as the rules are not broken.
* NOTE, A CHECKER MUST MAKE ALL POSSIBLE JUMPS IT CAN!

When a checker is clicked…

Check for legal moves:

List legalMoveBag

When a Checker is Clicked…

If(this.getCheckerColor() == player.getColor()){

If(this.getCheckerside() == BOTTOM){

If(board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getChecker() == null &&

board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getSquareColor() == board.darkColor){

legalMoveBag.add(board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getIndex());

}

}

}

forward

If(checker.getCheckerSide == BOTTOM){

//Check if Can move forward

If(checker.getRow – 1 < 0){return;}

//Check UpLeft

If(board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getChecker() == null &&

board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getSquareColor() == board.darkColor){

legalMoveBag.add(board[checker.getRow() – 1][checker.getIndex() – board.getRows() – 1].getIndex());

}

//Check UpRight

If(board[checker.getRow() – 1][checker.getIndex() – board.getRows() + 1].getChecker() == null &&

board[checker.getRow() – 1][checker.getIndex() – board.getRows() + 1].getSquareColor() == board.darkColor){

legalMoveBag.add(board[checker.getRow() – 1][checker.getIndex() – board.getRows() + 1].getIndex());

}

}

King Checker

if(Board[

Array[Board Rows x Board Cols]

Up Left: Current - Length of Row - 1

Up Right: Current - Length of Row + 1

Down Left: Current + Length of Row - 1

Down Right: Current + Length of Row + 1

To Check for Validity

UpLeft.isValid()?

isValid Checks if a space is free, a space is black,

Array For each Square on the Board.