Patrick Woodrum

Project 5 Extended Tic Tac Toe CpSc 2150 Section 001

Extended Tic Tac Toe Report

Requirements Analysis User Stories (Functional):

As a user, I should be able to:

-decide how many players will be playing the game (min:2,max:10) so that the game will be set up properly

-decide what character/letter I want to be represented by so that I have a team marker

-decide how many rows will be present on the game board so that I can customize the game

-decide how many columns will be present on the game board so that I can customize the game

-decide how many in a row a player needs to win the game so that I have control over how complex the game can be

-pick my row so that my token will be placed on that row

-pick my column so that my token will be placed on that column

-place my marker in the positions previously chosen

-view the entire board after each turn so that I can plan for next turn

-pick a new board position if the position I choose is out of range

-win the game and ask to play again

-end the game in a tie and ask to play again

-lose the game and ask to play again

Non-Functional:

As a system, it should be able to:

-This systems code was written in Java and must be able to be compiled and ran on Unix.

-The system will construct a board that is the size of the player’s choice

-The player will choose the amount of rows, columns, and number in a row to win

-The system will run until either a player wins, or there is a tie, then will be prompted to play again or not.

-The system will continue to run even if the user inputs an invalid integer.

-The system will ask the user for a new input if input was invalid.

-The system reads in the inputs from the players, and adequately assigns their move to the correct row and column.

-The system will update the gameboard after each turn to properly display where the tokens are placed.

-The system will check to see if there is a winner Vertically by having whatthe user inputted in a row and column.

-The system will check to see if there is a winner Horizontally by having what the user inputted in a row and column.

-The system will check to see if there is a winner Diagonally by having whatthe user inputted in a row and column.

-The system will display a message saying who won, if there is a winner.

-The system will display a message if a tie occurs.

-The system will prompt the user to play again or not.

**Test Case Description – Expected Output**

**Constructor Tests**

-Row Constructor Test

Input: Empty GameBoard of size 3 rows, 3 columns, numWin = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: Rows = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the constructor correctly creates a GameBoard with the   
 specified number of rows

-Column Constructor Test

Input: Empty GameBoard of size 3 rows, 3 columns, numWin = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: Columns = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the constructor correctly creates a GameBoard with the   
 specified number of columns

-Num to Win Constructor Test

Input: Empty GameBoard of size 3 rows, 3 columns, numWin = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: Num to Win = 3

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the constructor correctly creates a GameBoard with the   
 specified number to win equal to the constructor’s number using a getter

**CheckSpace Tests**

-Column Filled, Check Last Row

Input: GameBoard size [3][3]

X markers placed in [0][0] and [2][0]

O marker placed in [1][0]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | O |  |  |
| 2 | X |  |  |

Output: CheckSpace at [2][0] is false, space not available

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | O |  |  |
| 2 | X |  |  |

Reason: This tests that the checkSpace function can determine that the space in the last row and first column is filled and is not available

-No marker placed, checking empty spot

Input: GameBoard size [3][3]

No markers placed

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: CheckSpace at [0][0] is true, space is open

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the checkSpace function can determine that the space in the last first and first column is unfilled and available

-Last space filled, Check Last Row, Last Column

Input: GameBoard size [3][3]

X markers placed in [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Output: CheckSpace at [2][2] is false, space not available

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Reason: This tests that the checkSpace function can determine that the space in the last row and last column is filled and is not available

**CheckHorizontalWin Tests**

-Last Row win

Input: GameBoard size [3][3]

X markers placed in [2][0] and [2][1] and [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X | X | X |

Output: CheckHorizontalWin at [2][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X | X | X |

Reason: This tests that the checkHorizontalWin function is working properly in the last row of the board

-First Row win

Input: GameBoard size [3][3]

X markers placed in [0][0] and [0][1] and [0][2]

Checking space [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 |  |  |  |
| 2 |  |  |  |

Output: CheckHorizontalWin at [0][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the checkHorizontalWin function is working properly in the first row of the board

-Last Row win in larger board

Input: GameBoard size [5][5]

X markers placed in [4][1] and [4][2] and [4][3]

Checking space [4][3]

State of GameBoard:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  | X | X | X |  |

Output: CheckHorizontalWin at [4][3] is true

State of GameBoard:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |
| 4 |  | X | X | X |  |

Reason: This tests that the checkHorizontalWin function is working properly in the last row of the board, center orientation, and on a larger board

-Last Row, Bottom Left

Input: GameBoard size [3][3]

X markers placed in [2][0] and [2][1] and [2][2]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X | X | X |

Output: CheckHorizontalWin at [2][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X | X | X |

Reason: This tests that the checkHorizontalWin function is working properly in the last row of the board by checking the furthest left space

**CheckVerticalWin Tests**

-Middle Column win, testing 3 num win

Input: GameBoard size [3][3]

X markers placed in [0][1] and [1][1] and [2][1]

Checking space [2][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  | X |  |
| 1 |  | X |  |
| 2 |  | X |  |

Output: CheckVerticalWin at [2][1] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  | X |  |
| 1 |  | X |  |
| 2 |  | X |  |

Reason: This tests that the CheckVerticalWin function is working properly in the middle column of the board

-First Column win

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][0] and [2][0]

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | X |  |  |
| 2 | X |  |  |

Output: CheckVerticalWin at [0][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | X |  |  |
| 2 | X |  |  |

Reason: This tests that the CheckVerticalWin function is working properly in the first column of the board

-Last Column win, checking last space

Input: GameBoard size [3][3]

X markers placed in [0][2] and [1][2] and [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  |  | X |
| 2 |  |  | X |

Output: CheckVerticalWin at [2][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  |  | X |
| 2 |  |  | X |

Reason: This tests that the CheckVerticalWin function is working properly in the last row of the board

-First column win, last row check

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][0] and [2][0]

Checking space [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | X |  |  |
| 2 | X |  |  |

Output: CheckVerticalWin at [2][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 | X |  |  |
| 2 | X |  |  |

Reason: This tests that the CheckVerticalWin function is working properly in the first column of the board, last space available

**CheckDiagonalWin Tests**

-Forward win

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][1] and [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Output: CheckDiagonalWin at [2][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Reason: This tests that the CheckDiagonalWin function is working properly in the going forward on the final space of the board

-Backward win

Input: GameBoard size [3][3]

X markers placed in [0][2] and [1][1] and [2][0]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Output: CheckDiagonalWin at [2][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Reason: This tests that the CheckDiagonalWin function is working properly in the going backwards in the first column check of the board

-Forward win checking top left now

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][1] and [2][2]

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Output: CheckDiagonalWin at [0][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Reason: This tests that the CheckDiagonalWin function is working properly in the by checking the top left and going down

-Top right going backward

Input: GameBoard size [3][3]

X markers placed in [0][2] and [1][1] and [2][0]

Checking space [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Output: CheckDiagonalWin at [0][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Reason: This tests that the CheckDiagonalWin function is working properly in the first row of the board going backward

-Bottom left backward

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][1] and [2][0]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Output: CheckDiagonalWin at [2][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  | X |  |
| 2 | X |  |  |

Reason: This tests that the CheckDiagonalWin function is working properly in the last row of the board checking backward win

-Bottom Right forward

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][1] and [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Output: CheckDiagonalWin at [2][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  | X |

Reason: This tests that the CheckDiagonalWin function is working properly in the bottom right of the board going forward

-No Diagonal win, not enough pieces

Input: GameBoard size [3][3]

X markers placed in [0][0] and [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Output: CheckHorizontalWin at [1][1] is false

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Reason: This tests that the CheckDiagonalWin function is working properly in the by throwing a false when the diagonalwin is not complete

**CheckDraw Tests**

-Top left draw

Input: GameBoard size [3][3]

X markers placed in every spot on the board

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Output: CheckDraw at [0][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Reason: This tests that the CheckDraw function is working properly in the first spot of the board

-Bottom left draw

Input: GameBoard size [3][3]

X markers placed in every spot on the board

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Output: CheckDraw at [2][0] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Reason: This tests that the CheckDraw function is working properly in the first column, last row of the board

-Top right draw

Input: GameBoard size [3][3]

X markers placed in every spot on the board

Checking space [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Output: CheckDraw at [0][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Reason: This tests that the CheckDraw function is working properly in the first row and last column of the board

-Bottom right draw

Input: GameBoard size [3][3]

X markers placed in every spot on the board

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Output: CheckDraw at [2][2] is true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X | X | X |
| 1 | X | X | X |
| 2 | X | X | X |

Reason: This tests that the CheckDraw function is working properly in the last spot of the board

**WhatsAtPos Tests**

-Checking center of board

Input: GameBoard size [3][3]

X markers placed at [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Output: WhatsAtPos at [1][1] returns player ‘X’

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Reason: This tests that the WhatsAtPos function is working properly by returning the character that is at the specified position in the center of the board

-Checking center of board returning different character

Input: GameBoard size [3][3]

X markers placed at [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Output: WhatsAtPos at [1][1] returns player ‘O’

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Reason: This tests that the WhatsAtPos function is working properly by returning the character that is at the specified position in the center of the board

-Checking top left of board

Input: GameBoard size [3][3]

X markers placed at [0][0]

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: WhatsAtPos at [0][0] returns player ‘X’

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the WhatsAtPos function is working properly by returning the character that is at the specified position on the board

-Checking bottom left of board

Input: GameBoard size [3][3]

X markers placed at [2][0]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X |  |  |

Output: WhatsAtPos at [2][0] returns player ‘X’

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X |  |  |

Reason: This tests that the WhatsAtPos function is working properly by returning the character that is at the specified position on the board

-Checking bottom right of board

Input: GameBoard size [3][3]

X markers placed at [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Output: WhatsAtPos at [2][2] returns player ‘X’

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Reason: This tests that the WhatsAtPos function is working properly by returning the character that is at the specified position on the board

**IsPlayerAtPos Tests**

-Checking top left of board

Input: GameBoard size [3][3]

X markers placed at [0][0]

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: IsPlayerAtPos at [0][0] returns true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the IsPlayerAtPos function is working properly by returning true if that player is at the specified position on the board

-Checking center of board

Input: GameBoard size [3][3]

X markers placed at [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Output: IsPlayerAtPos at [1][1] returns true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | X |  |
| 2 |  |  |  |

Reason: This tests that the IsPlayerAtPos function is working properly by returning true if that player is at the specified position on the board

-Checking bottom right of board

Input: GameBoard size [3][3]

X markers placed at [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Output: IsPlayerAtPos at [2][2] returns true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Reason: This tests that the IsPlayerAtPos function is working properly by returning true if that player is at the specified position on the board

-Checking center false

Input: GameBoard size [3][3]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: IsPlayerAtPos at [1][1] returns false

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the IsPlayerAtPos function is working properly by returning false if that player is not at the specified position on the board

-Checking top left of board

Input: GameBoard size [3][3]

O markers placed at [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Output: IsPlayerAtPos at [1][1] returns true

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Reason: This tests that the IsPlayerAtPos function is working properly by returning false if that player is not at the specified position on the board

**PlaceMarker Tests**

-Checking top left of board

Input: GameBoard size [3][3]

X markers placed at [0][0]

Checking space [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Output: new marker placed at [0][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 | X |  |  |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the placeMarker function is working properly by placing the specified character at the location given on the board

-Checking top right of board

Input: GameBoard size [3][3]

X markers placed at [0][2]

Checking space [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  |  |  |
| 2 |  |  |  |

Output: new marker placed at [0][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  | X |
| 1 |  |  |  |
| 2 |  |  |  |

Reason: This tests that the placeMarker function is working properly by placing the specified character at the location given on the board

-Checking bottom left of board

Input: GameBoard size [3][3]

X markers placed at [2][0]

Checking space [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X |  |  |

Output: new marker placed at [2][0]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 | X |  |  |

Reason: This tests that the placeMarker function is working properly by placing the specified character at the location given on the board

-Checking bottom right of board

Input: GameBoard size [3][3]

X markers placed at [2][2]

Checking space [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Output: new marker placed at [2][2]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  |  |  |
| 2 |  |  | X |

Reason: This tests that the placeMarker function is working properly by placing the specified character at the location given on the board

-Checking center of board with new character

Input: GameBoard size [3][3]

O markers placed at [1][1]

Checking space [1][1]

State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Output: new marker placed at [1][1]

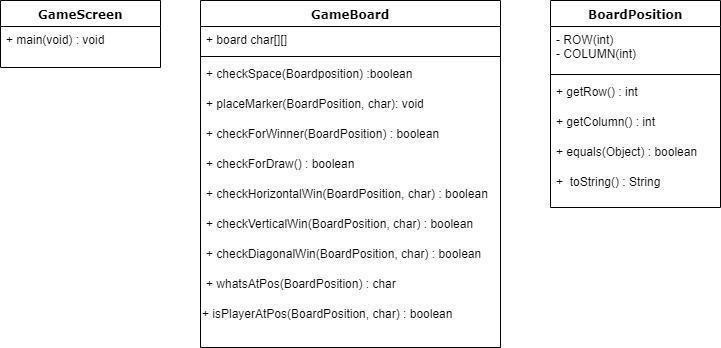
State of GameBoard:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 0 | 1 | 2 |
| 0 |  |  |  |
| 1 |  | O |  |
| 2 |  |  |  |

Reason: This tests that the placeMarker function is working properly by placing the specified character at the location given on the board

Design

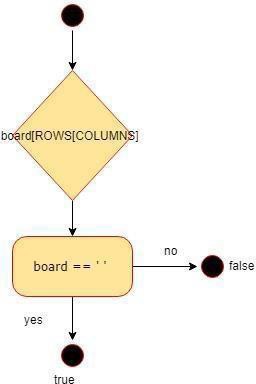
UML Class Diagrams:

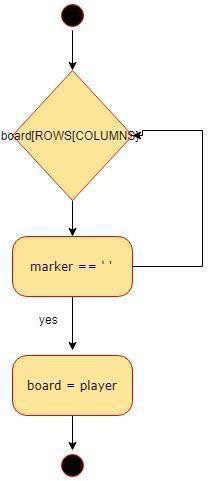


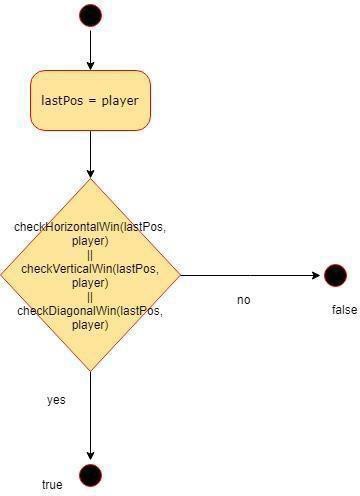
UML Activity Diagrams:

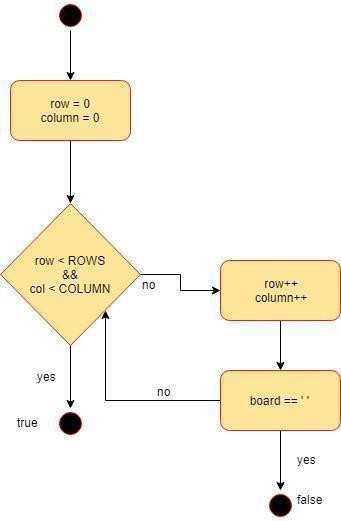
GameBoard.java

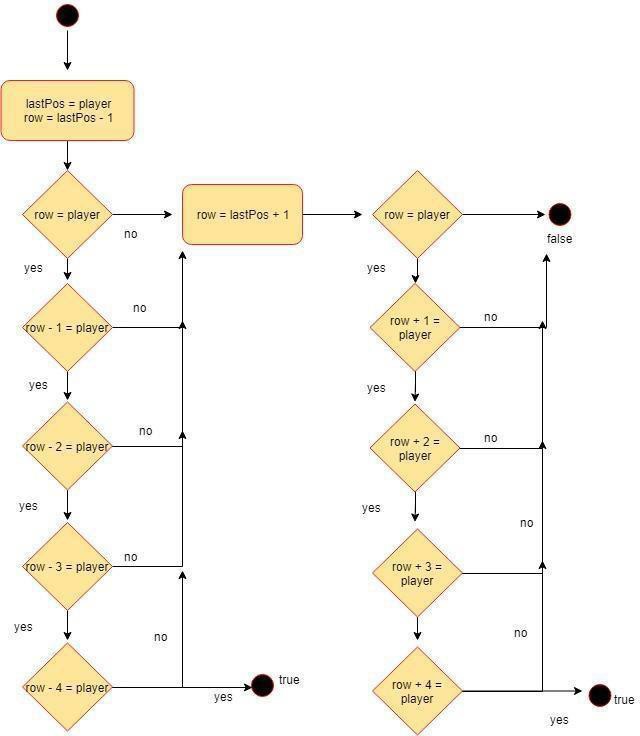
default boolean checkSpace(BoardPosition pos)

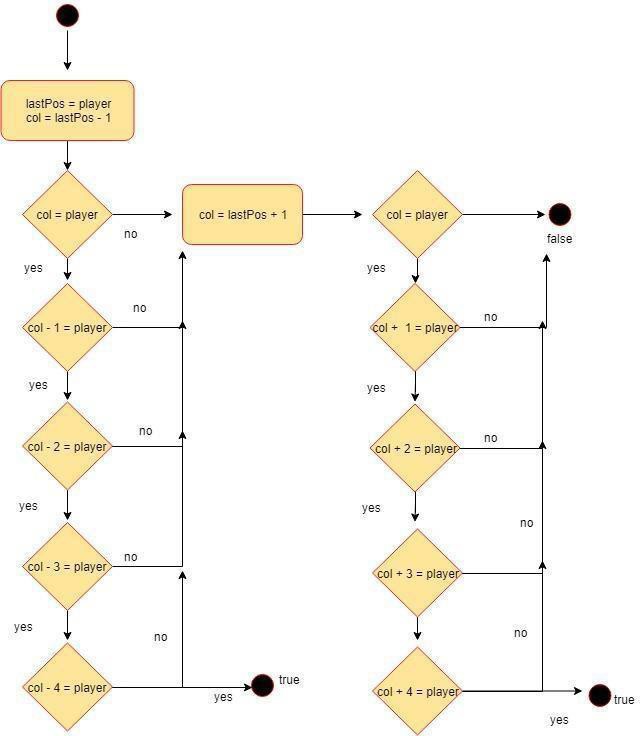


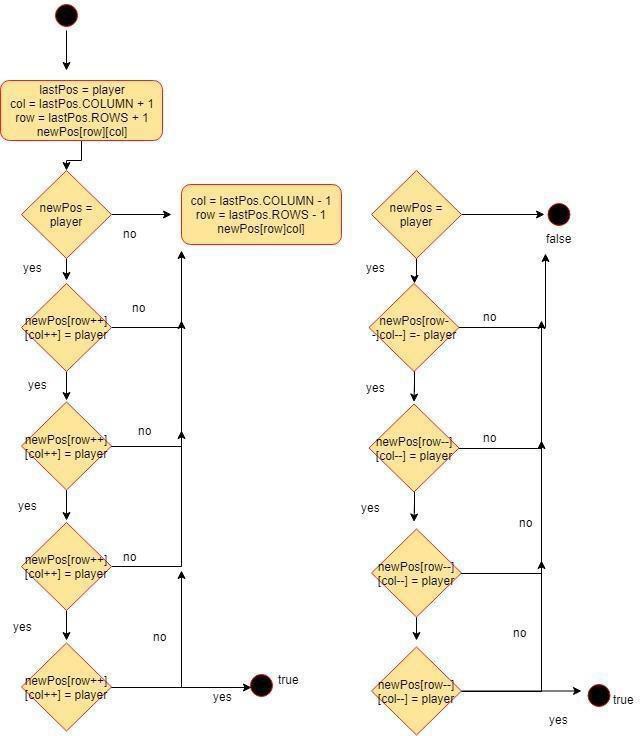
Public void placeMarker(BoardPosition marker, char player)

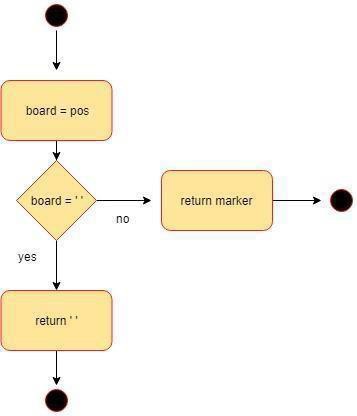
default boolean checkForWinner(BoardPosition lastPos)

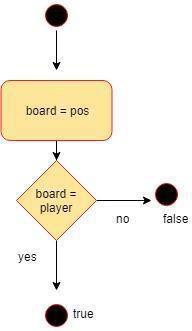
default boolean checkForDraw()

default boolean checkHorizontalWin(BoardPosition lastPos, char player)

default boolean checkVerticalWin(BoardPosition lastPos, char player)

default boolean checkDiagonalWin(BoardPosition lastPos, char player)

Public char whatsAtPos(BoardPosition pos)

default boolean isPlayerAtPos(BoardPosition pos, char player)

GAMESCREEN: public static void main(String [] args)

