

Home Work 3 Lab Report

CPSC 2150
October 28, 2018
Thomas Griffith

Functional Requirements:

As a player, I can choose a number so my token will be on the board

As a player, I can type in n or N when the game is over to leave the game

As a player, I can type in y or Y when the game is over to play again

As a player, I can line up my token so that it can be diagonal to win

As a player, I can line up my token so that it can be horizontal to win

As a player, I can line up my token so that it can be vertical to win

Users are able to play multiple games if they desired it

As a player, I can set the row amount, so I can play a bigger game

As a player, I can set the column amount, so I can play a bigger game

When a game is done, a user can rescale board, so he can change the board size

When the game is starting, a user can choose the number of players, if he wants to.

When the game is starting, a user can choose the game board class so it can be more memory effective

When the game is starting, a user can choose the game board class so it can be faster

Users are allowed controlled selection of where to drop their token on their turn

Non-Functional Requirements:

The system must run on java systems

The program should be user friendly and aesthetically pleasing

The program instructions should be clear and easy to understand

Must allow the quick ability to change the code if needed to mutate game play

Must provide extensive documentation for user and other programmers

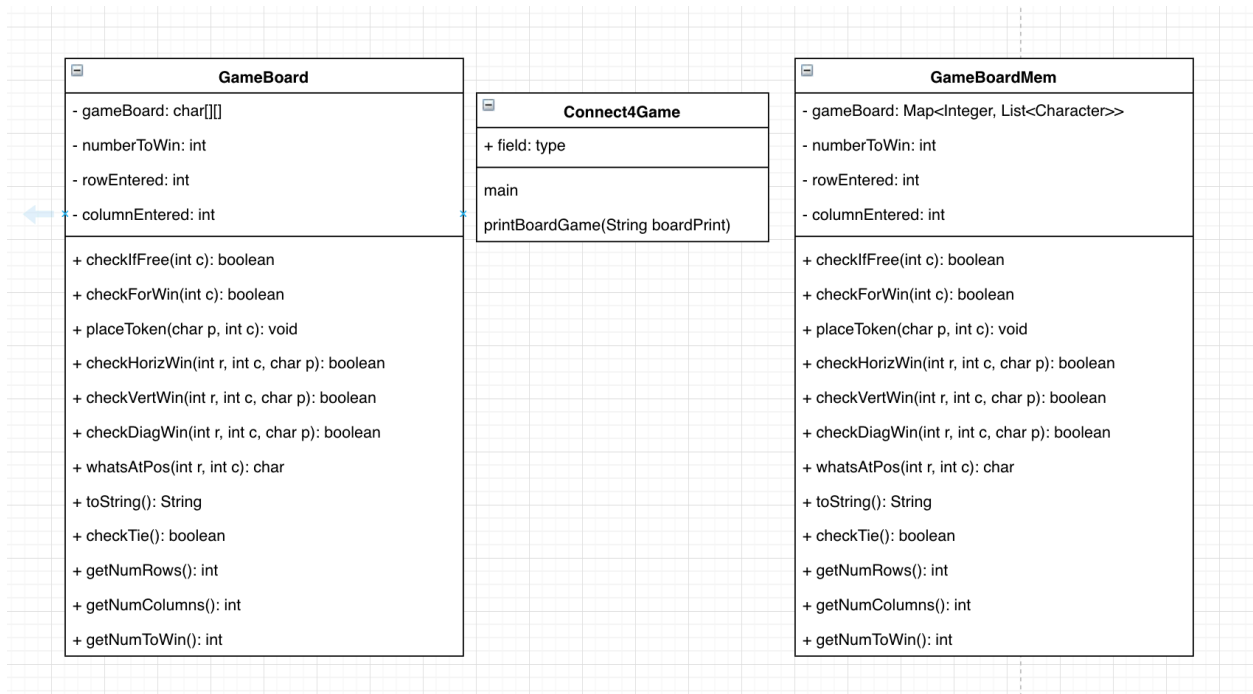
Must be able to operate with only two players

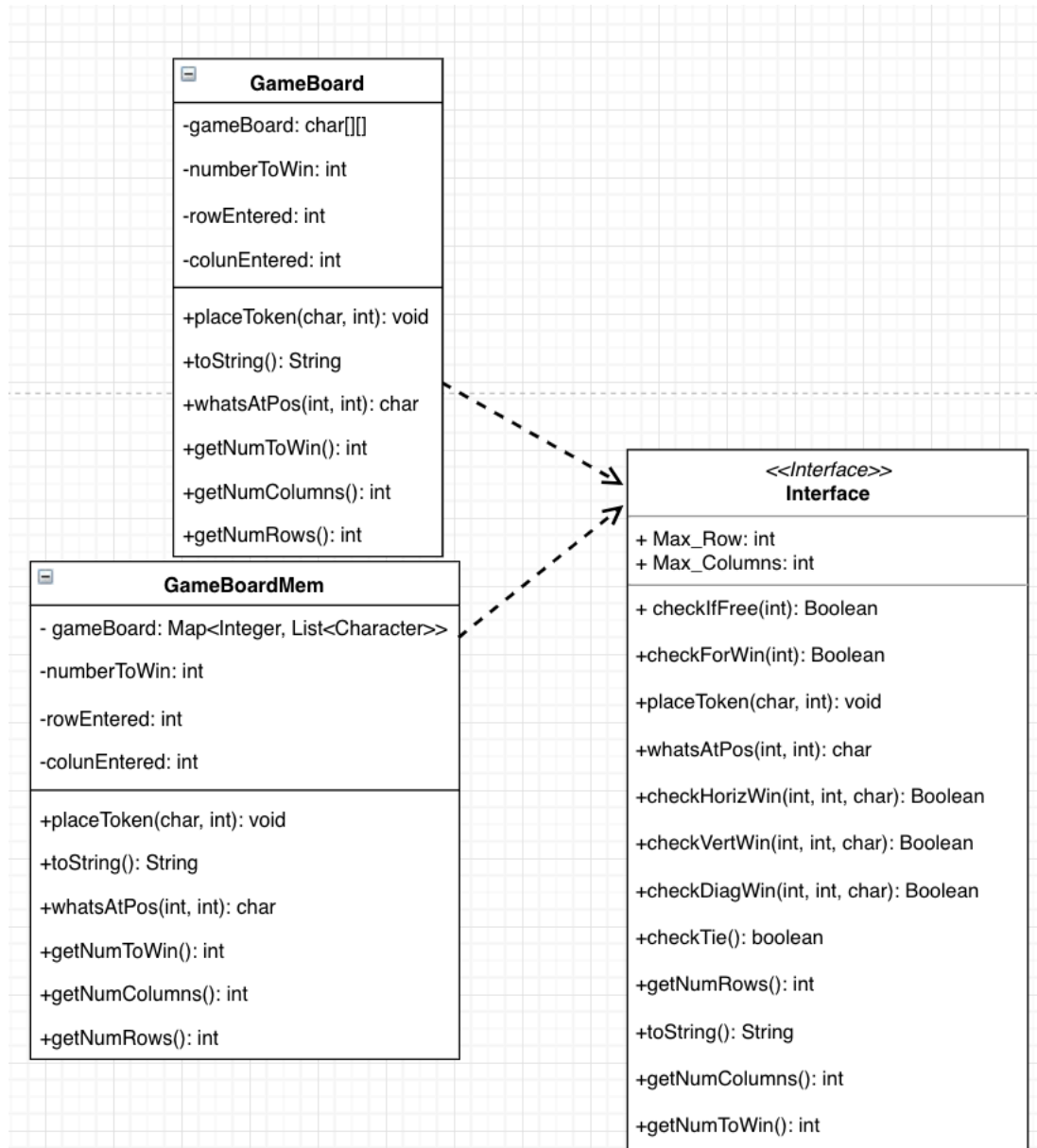
Must be able to only have one game going on at a time

Platform requires the ability for java commands and operations

Must use the interface for the gameboard class

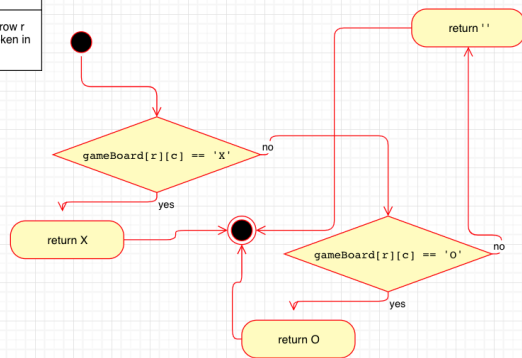
Design





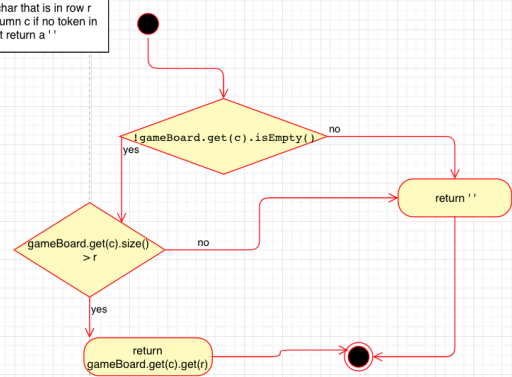
char whatsAtPos
param (int r, int c)
return char that is in row r and column c if no token in the spot return a ' '

Fast Imp



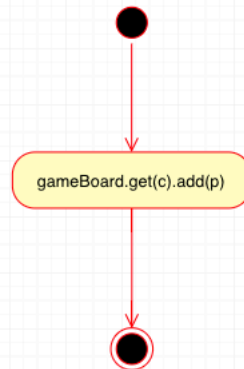
char whatsAtPos
param (int r, int c)
return char that is in row r and column c if no token in the spot return a ' '

Mem Eff. Imp



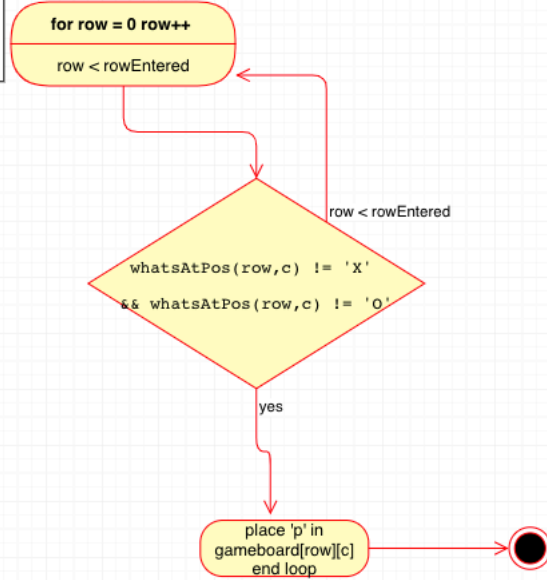
	void placeToken
param (char p, int c)	
places token p in column c placed in lowest available row cannot be called if column is full	

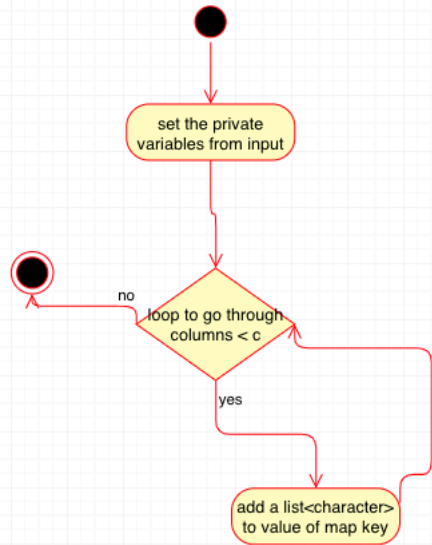
mem eff. imp



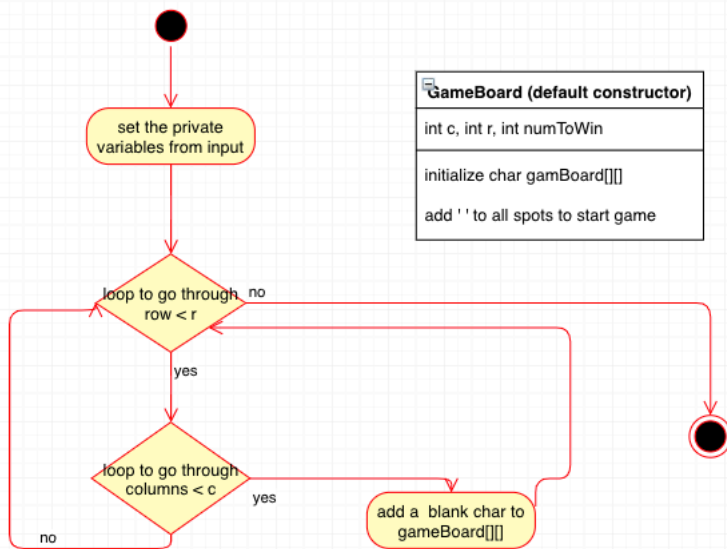
	void placeToken
param (char p, int c)	
places token p in column c placed in lowest available row cannot be called if column is full	

fast imp





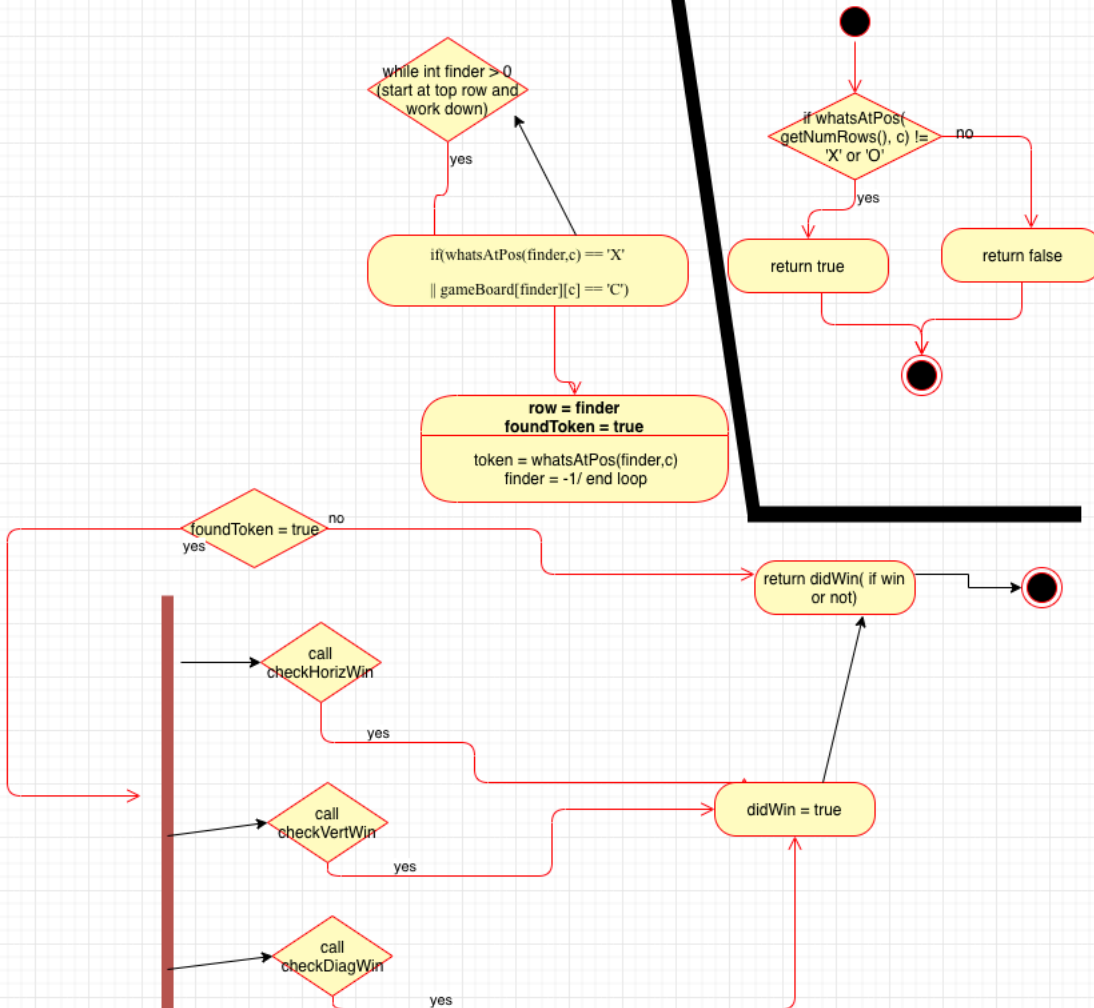
GameBoardMem (default constructor)
int c, int r, int numToWin
initialize map gameBoardMem
add a empty list to map

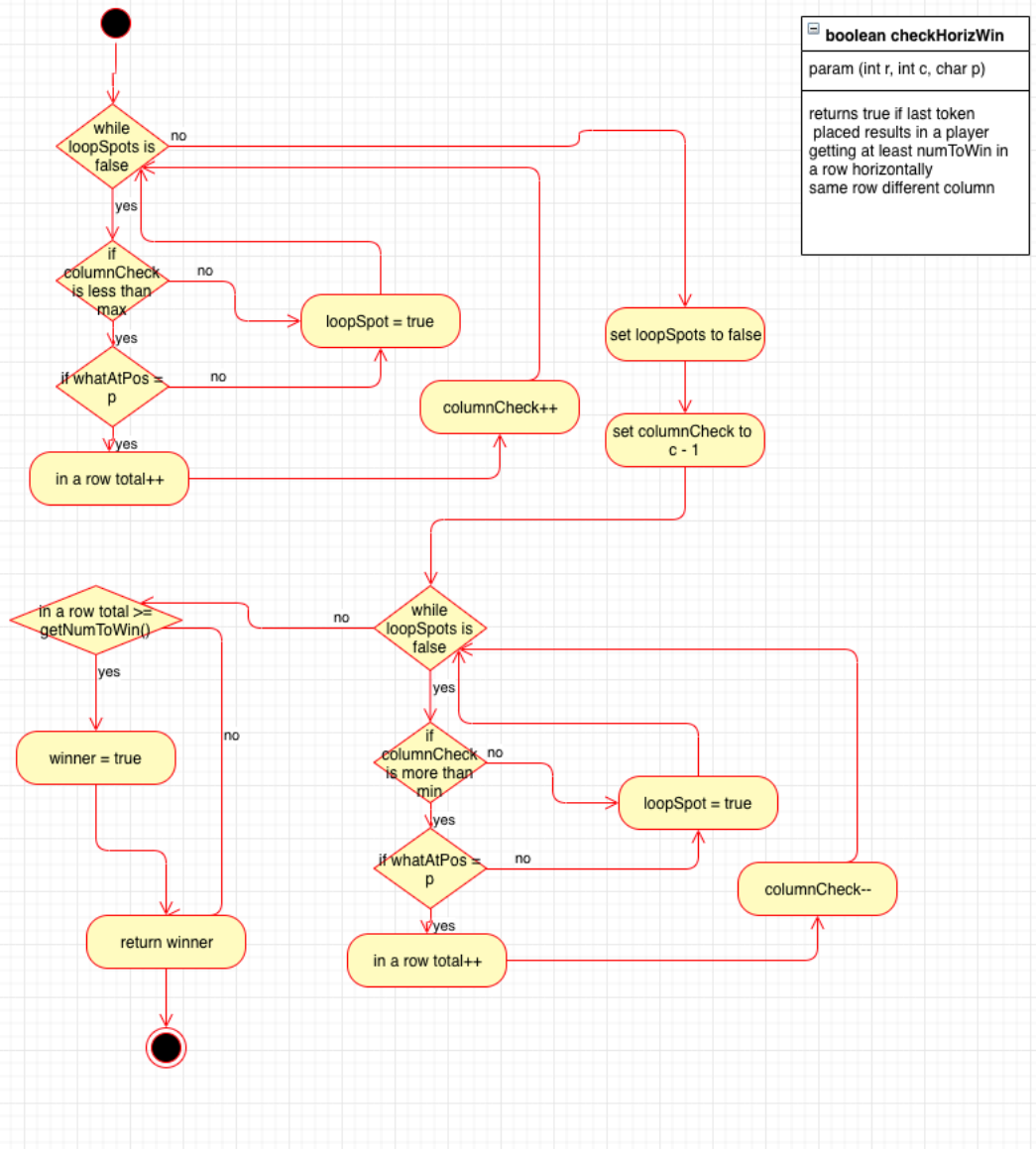



GameBoard (default constructor)
int c, int r, int numToWin
initialize char gamBoard[][]
add ' ' to all spots to start game

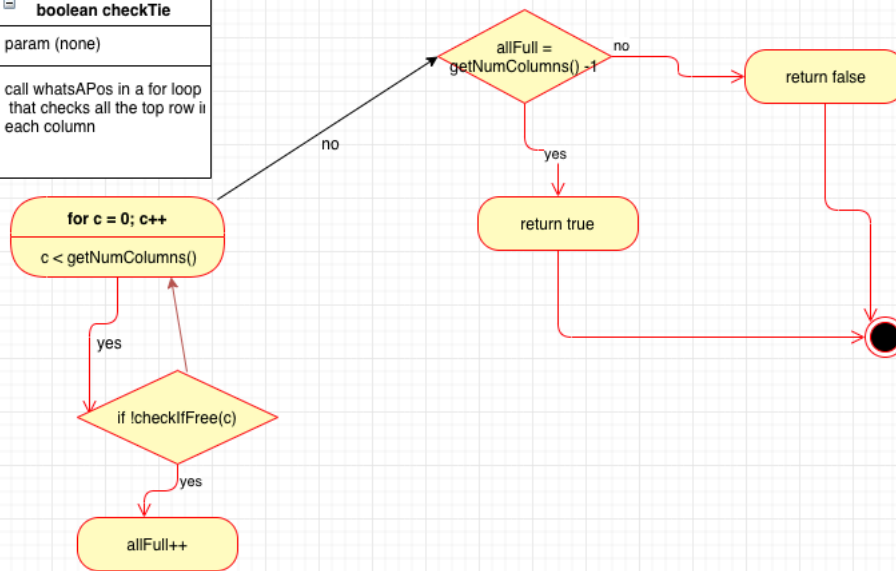
boolean checkForWin
param (int c)
returns true if last token in (c) resulted in the player get numToWin in a row horiz diag, or vert. (call those three other functions)

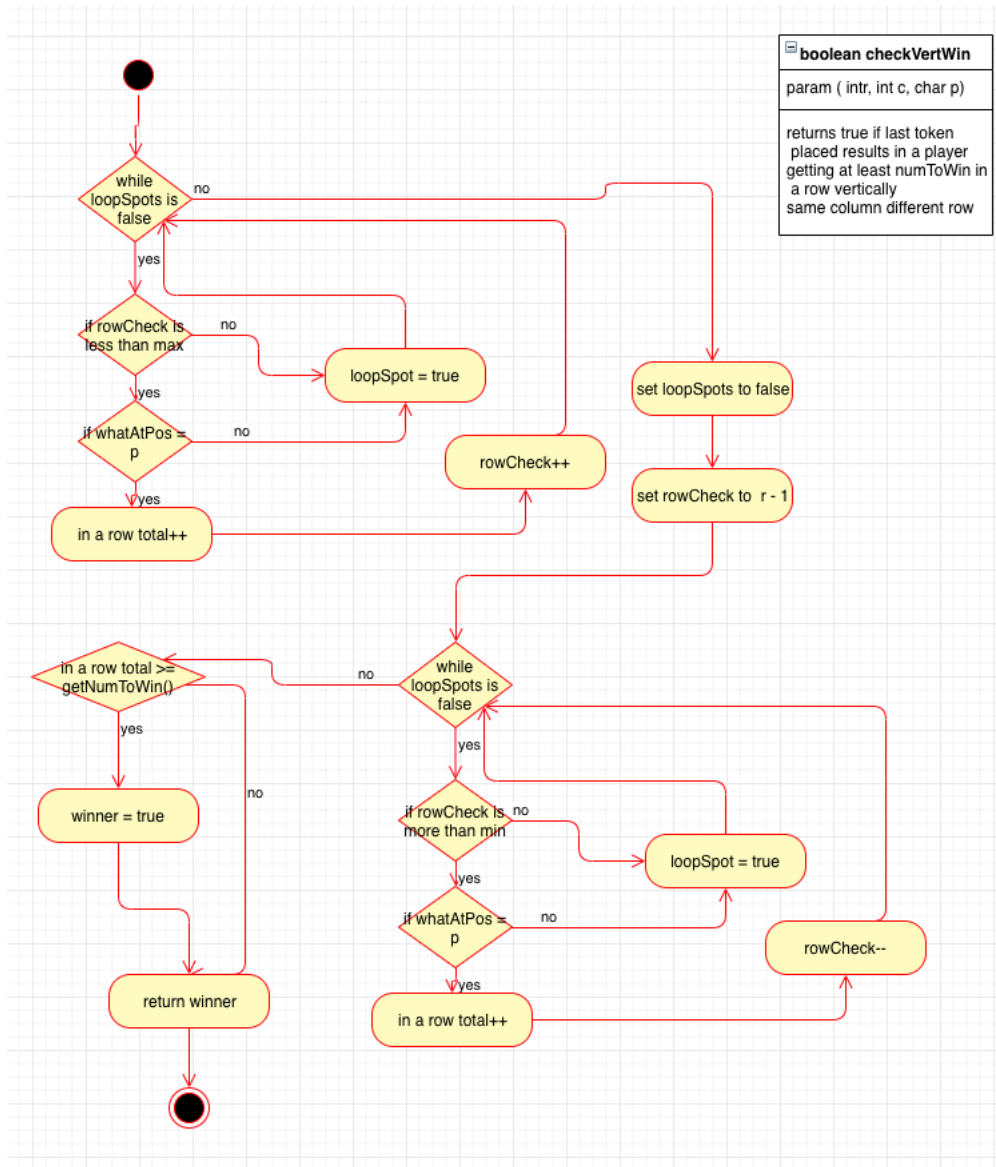
boolean checkIfFree
param (int c)
returns true if column c is no full



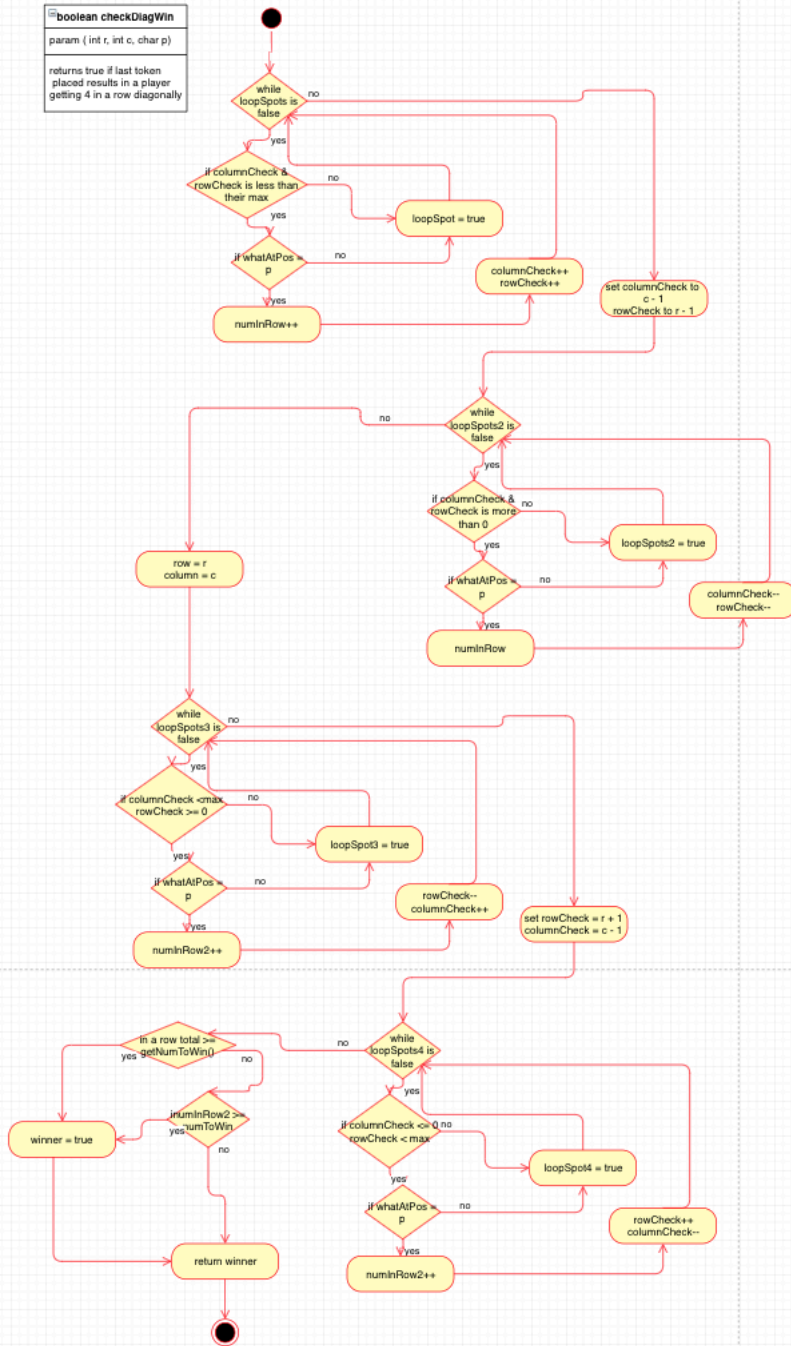


 boolean checkTie
param (none)
call whatsAPos in a for loop that checks all the top row ii each column





boolean checkDiagWin
 param (int r, int c, char p)
 returns true if last token
 placed results in a player
 getting 4 in a row diagonally



Testing:

To test my code, I tested all possible ways to win vertically, horizontally, diagonally on the borders and edge cases and normal places and if the last placed token was in the middle or end or the order. Also I checked if a tie was determined. I tested these winning options on both the memory efficient and fast implementation of the interface. I checked if I can add tokens to full columns too. Also I checked the amount of players, rows and columns all were properly used and displayed. Also, when you win or tie, you can restart and reselect all the options given i.e. number of players, type of board and size. Also I tried out of bounds numbers for the columns and row and number of players in the beginning and out of bound columns in the game. Also you can choose to play again or exit tested for each board type.

---- Horizontal | Vertical ^Diagonal (y,Y) play again (n,N)stop playing

Deployment:

Type make

To compile

Then make run