CPSC 2150 Project Report

Skylar Hubbarth

Requirements Analysis

Functional Requirements:

1. As a player, I can enter the column number so that I can place my token in the spot I desire.
2. As a player, I can play the game again after a round is over so that I can continue playing the game with a new board.
3. As a player, I can stop the game after a round is over so that I can stop the program and don't have to play any longer.
4. As a player, I can play the game with my friend on one computer so that we can play a game together and see who wins.
5. As a player, I can play the game by myself so that I can test out how the game works.
6. As a player, I can switch back and forth between player X and player O so we can each have a turn.
7. As a player, I can fill up the entire board so that the game will end in a tie.
8. As a player, I can align my tokens in a horizontal line 5 tokens long so that I can win the game.
9. As a player, I can align my tokens in a vertical line 5 tokens high so that I can win the game.
10. As a player, I can align my tokens in a diagonal line 5 tokens long so that I can win the game.
11. As a player, I can place my tokens on top of my opponent's tokens so that I can cut off their tokens and keep them from winning.
12. As a player, I can see where previous tokens were placed on the board so that I can know where I'm able to place my tokens next.
13. As a player, I can see my previous tokens that match my respective player symbol (X or O) so that I can connect five of them and win the game.
14. As a player, I can see my opponent's previous tokens that match their respective player symbol (X or O) so that I can see when they are about to win the game and try to stop them.
15. As a player, I can pick again if I pick an unavailable column, so I don't lose my turn.
16. As a player, I can pick again if I pick a column that does not exist, so I don't lose my turn.

Non-Functional Requirements

1. The game will be played in a terminal.
2. The program will be started in a terminal.
3. The user cannot enter a column number less than 0.
4. The user cannot enter a column number greater than 8.
5. The user cannot place a token in a column that has been selected 6 times in one round.
6. The user will select their column number by entering a number (0-8) on the keyboard and pressing enter in the terminal.
7. The program will be written in java.
8. The board size is of size 6 x 9.
9. X always goes first.
10. Position 0,0 is at the bottom left of the game board.

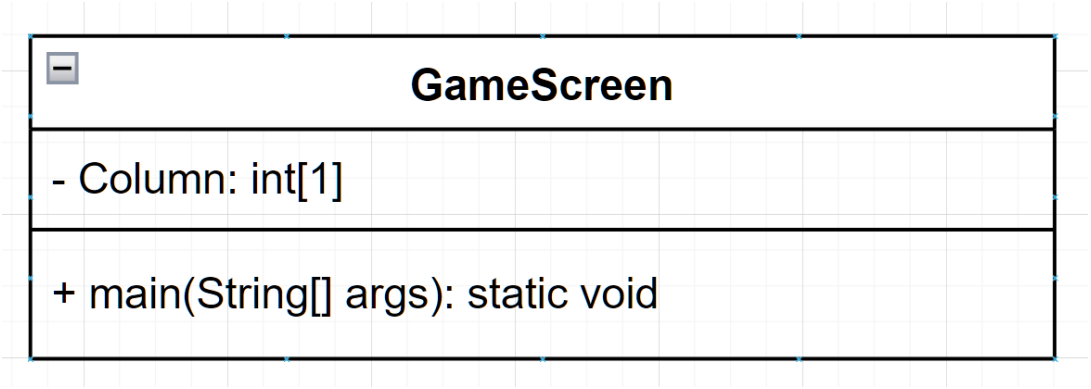
Deployment Instructions

Details in Projects 2-5.

System Design

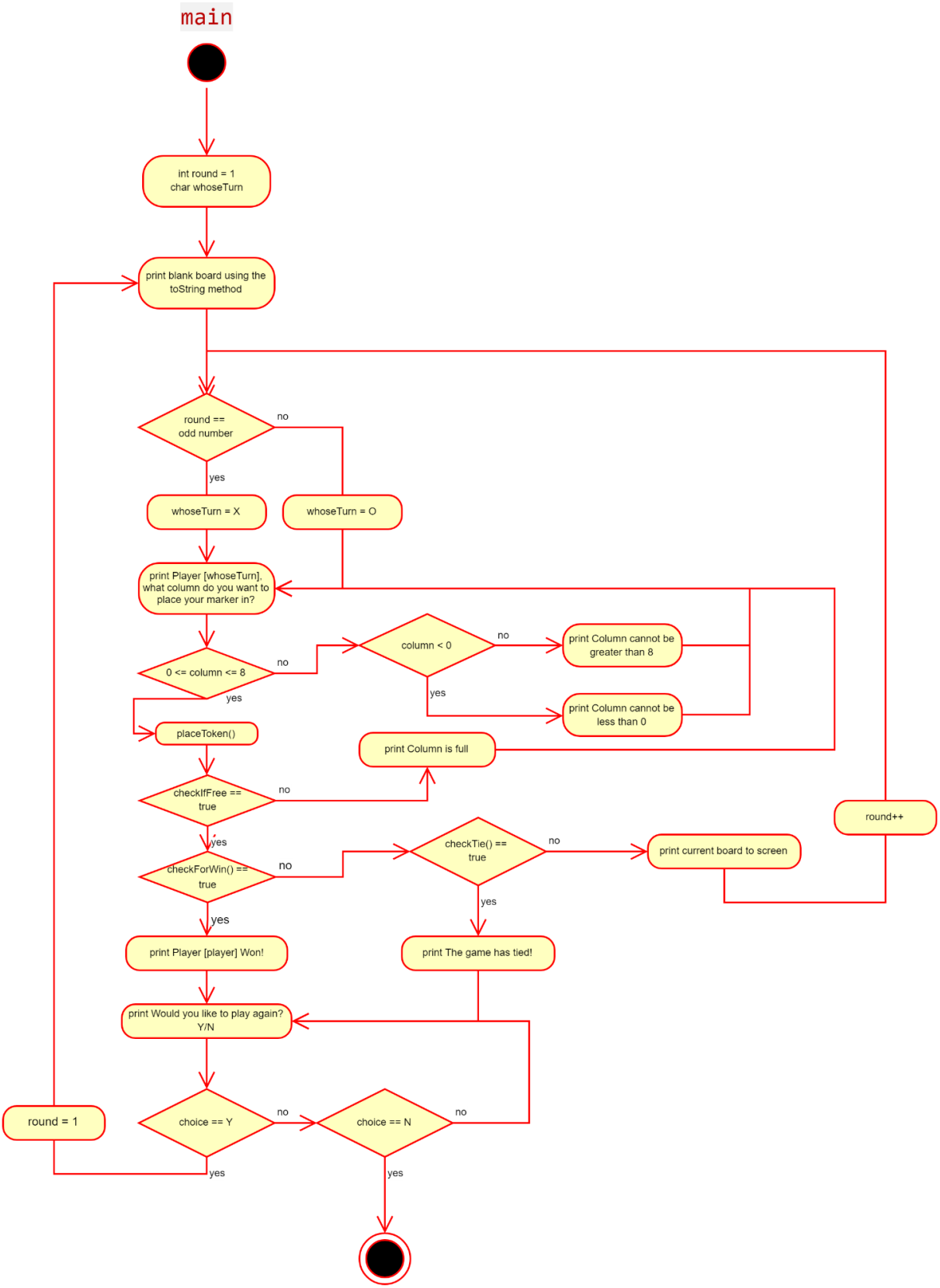
Class 1: `GameScreen`

Class diagram



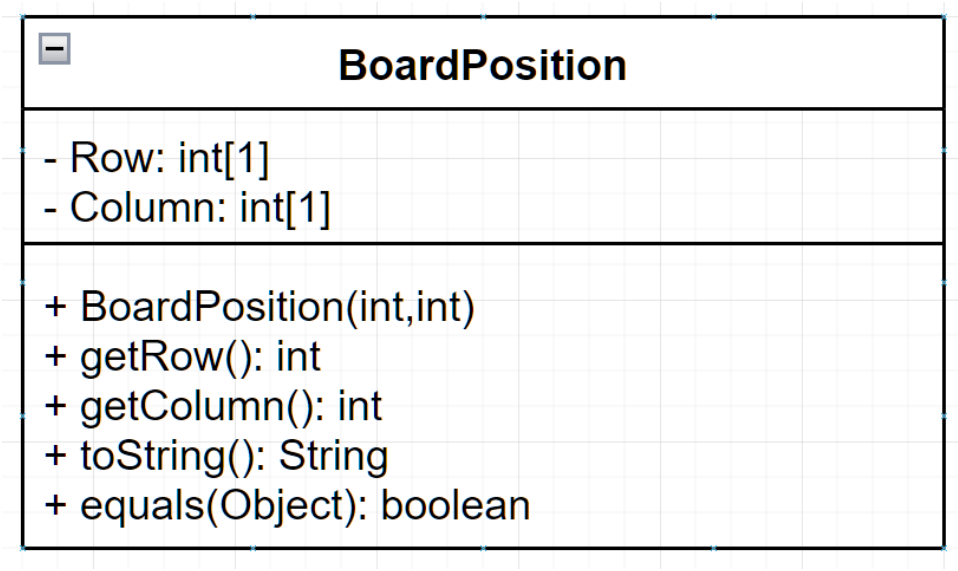
Activity diagrams

`main()`


main
int round = 1
char whoseTurn
print blank board using the toString method
round == odd number
no
yes
whoseTurn = X
whoseTurn = O
print Player [whoseTurn], what column do you want to place your marker in?
0 <= column <= 8
no
yes
column < 0
no
yes
print Column cannot be greater than 8
print Column cannot be less than 0
placeToken()
print Column is full
checkIfFree == true
no
yes
checkForWin() == true
no
yes
checkTie() == true
no
yes
print current board to screen
round++
print Player [player] Won!
print The game has tied!
print Would you like to play again? Y/N
round = 1
choice == Y
no
yes
choice == N
no
yes

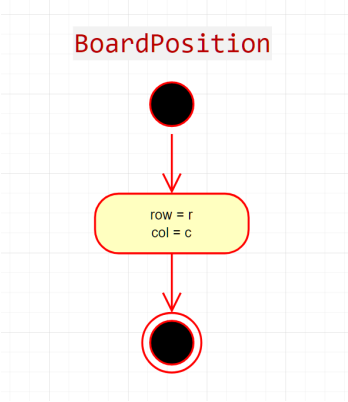
Class 2: `BoardPosition`

Class diagram

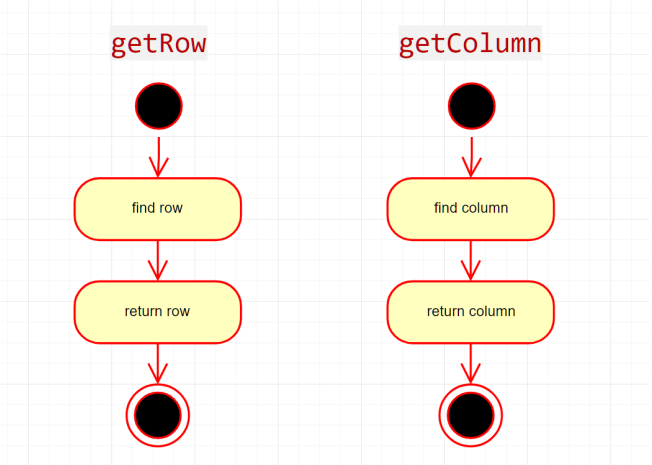


Activity diagrams

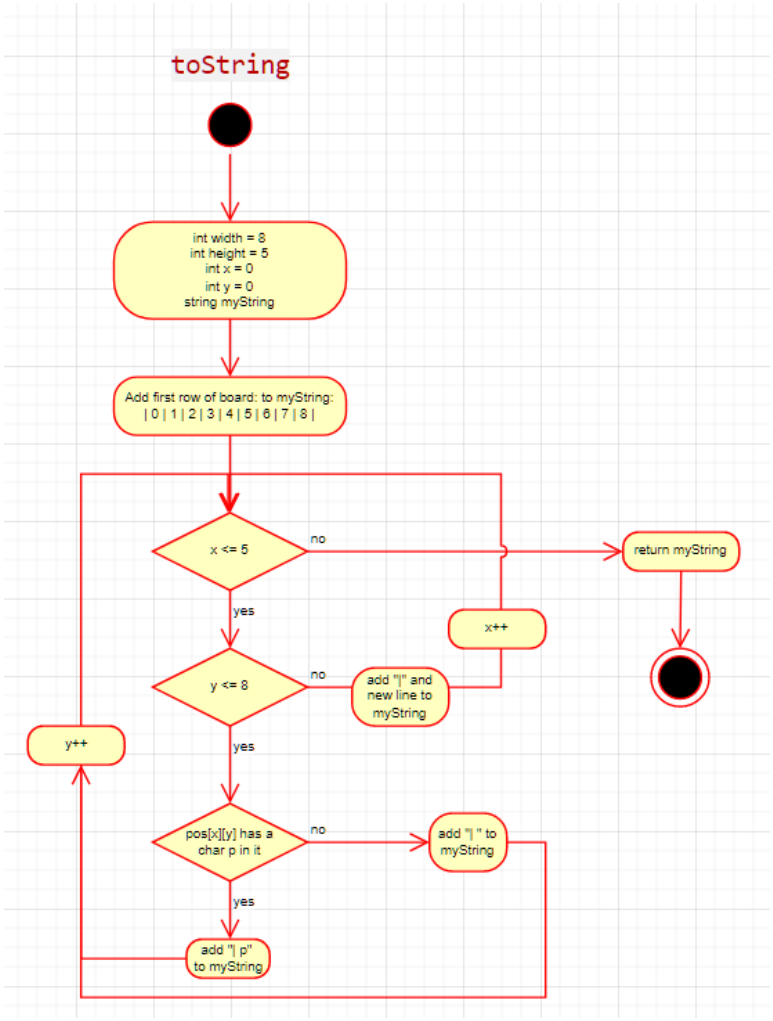
`BoardPosition`



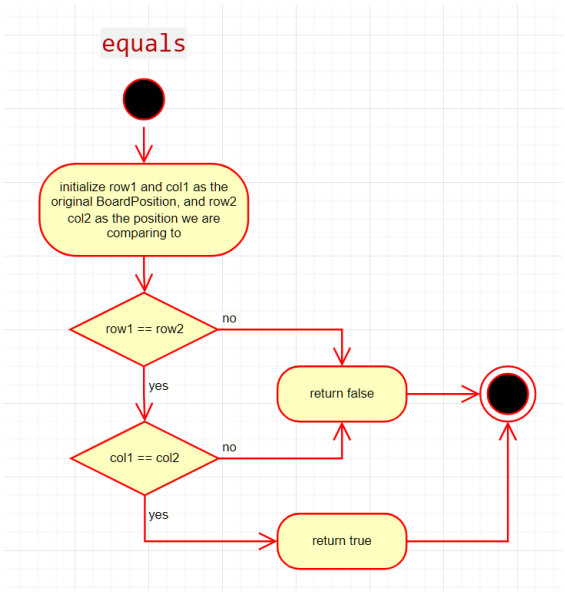
`getRow` `getColumn`



toString

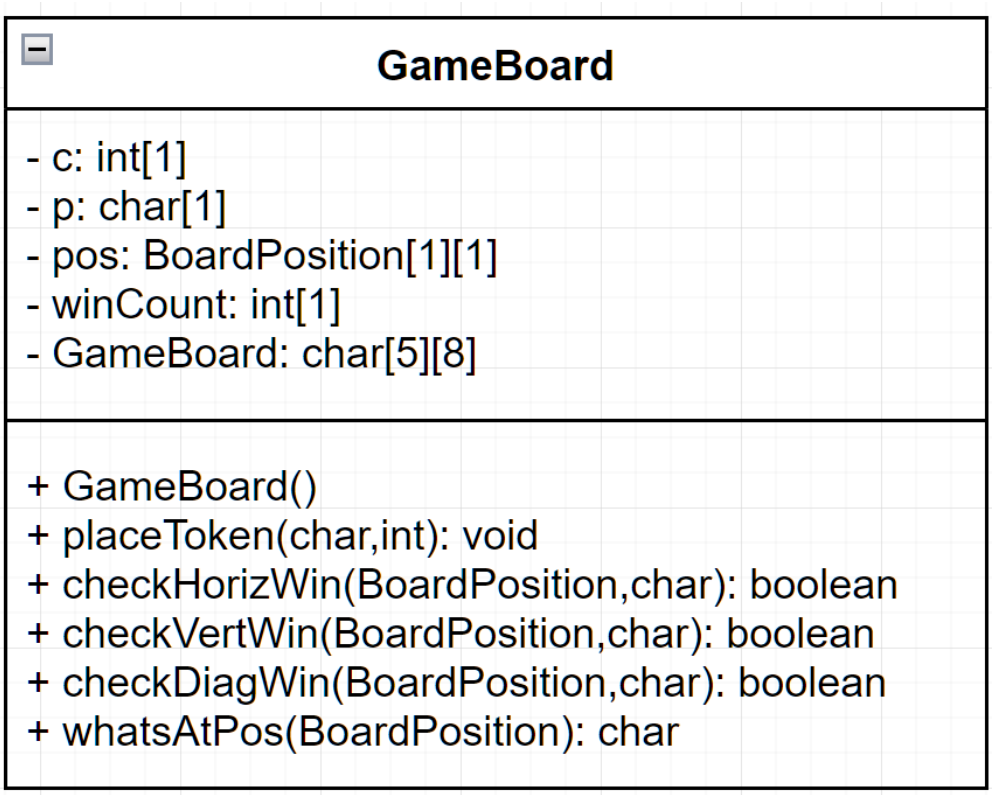


equals



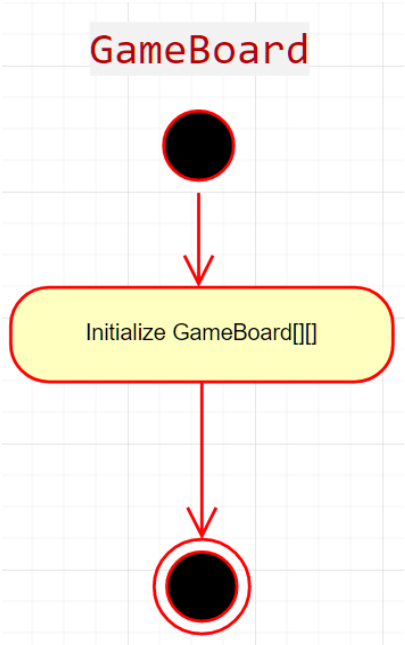
Class 3.0: `GameBoard`

Class diagram

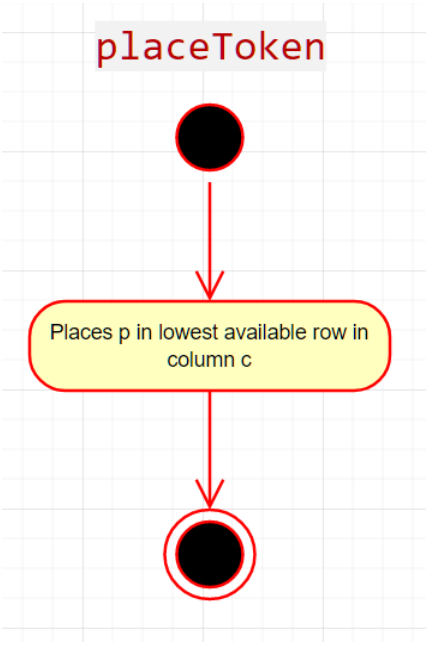


Activity diagrams

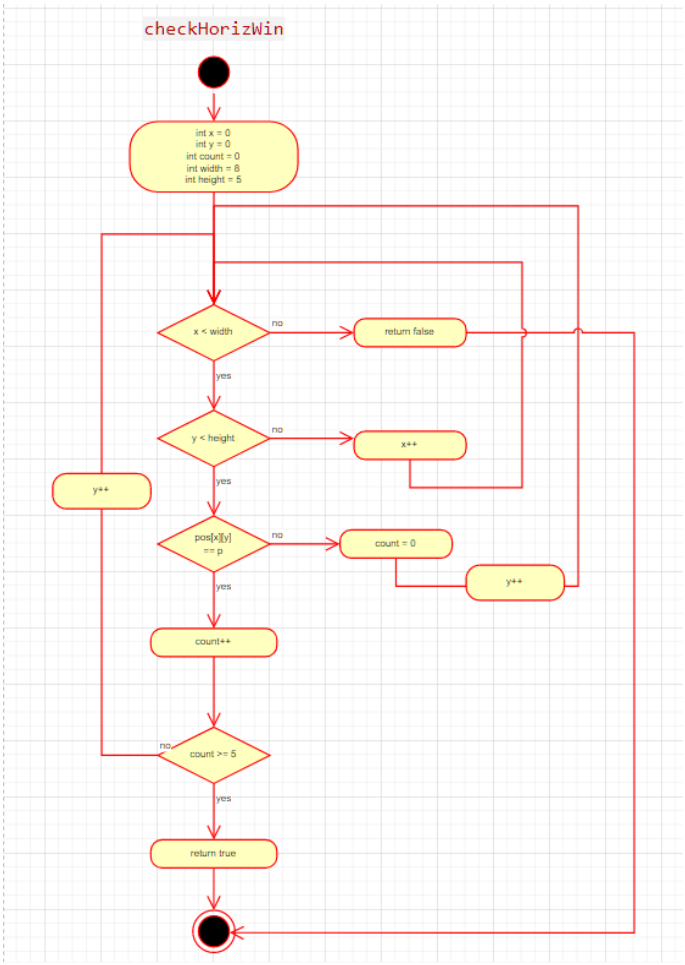
`GameBoard`



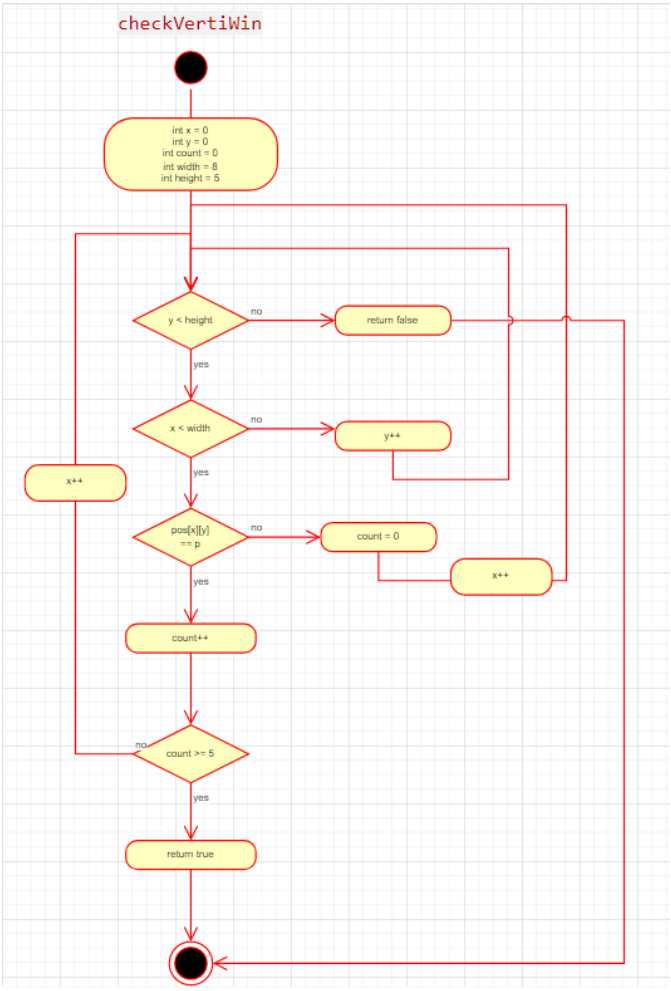
placeToken



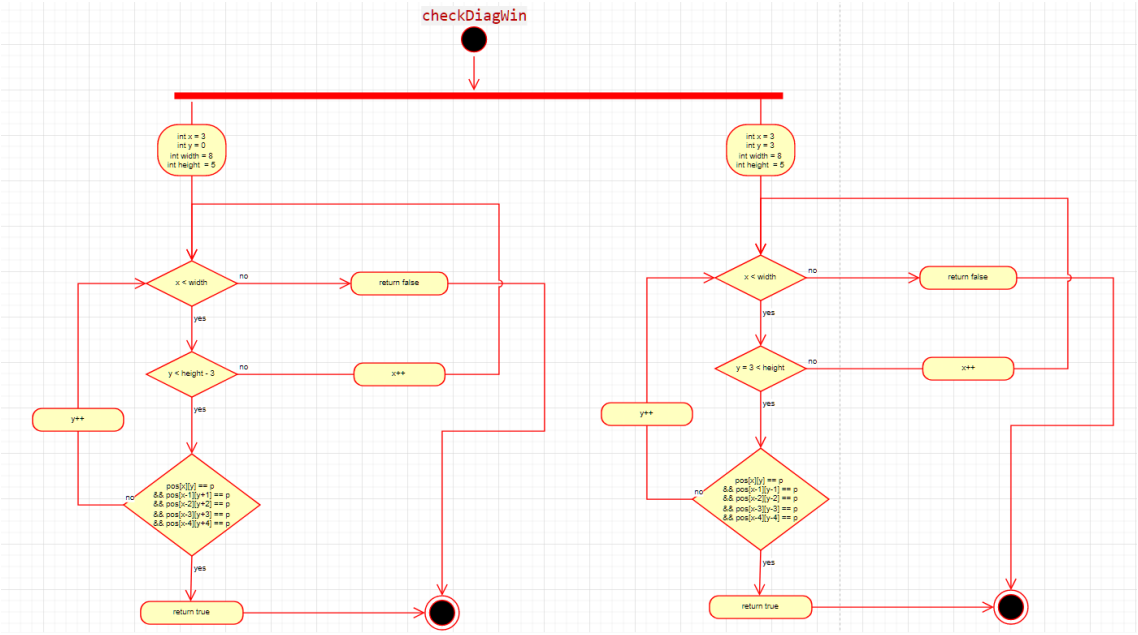
checkHorizWin



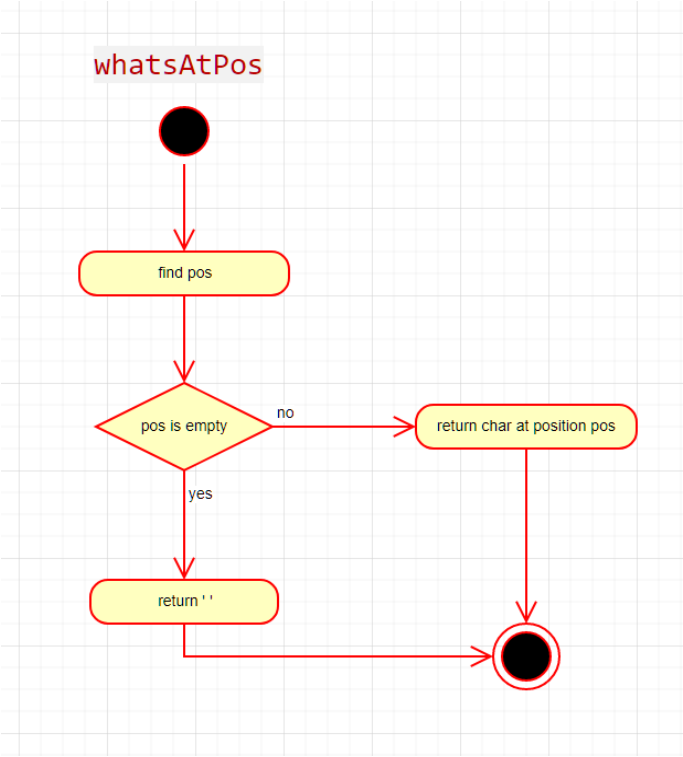
checkVertiWin



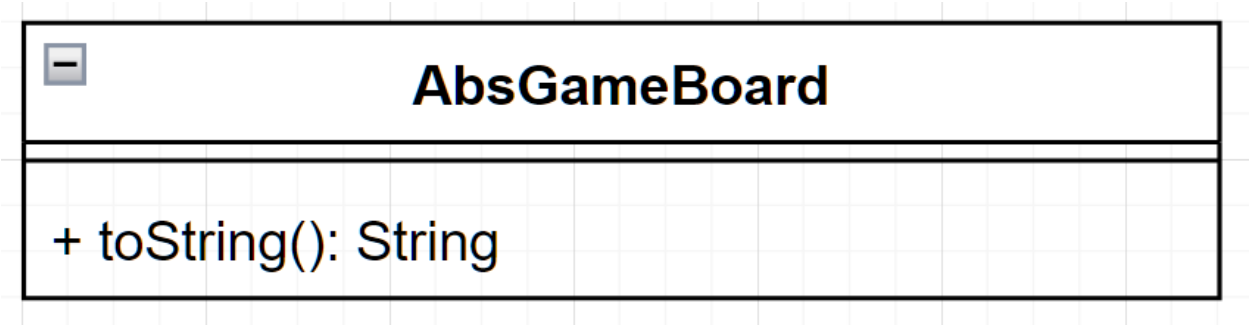
checkDiagWin



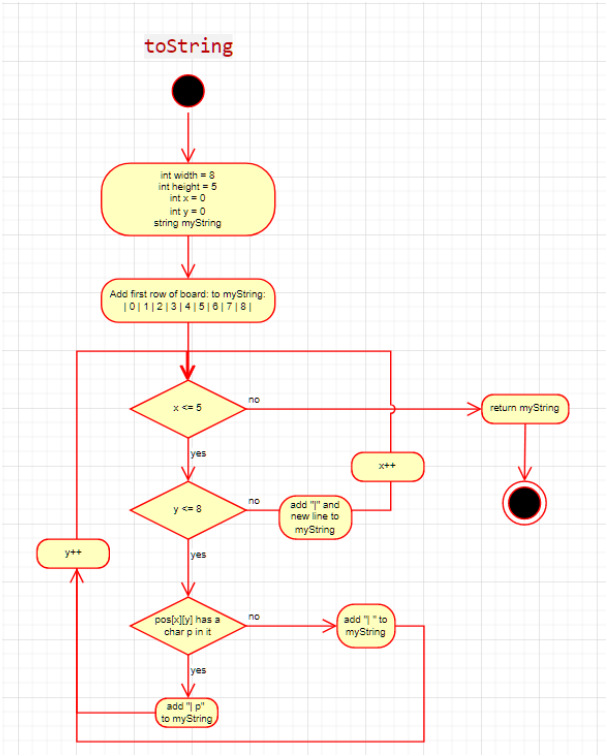
`whatsAtPos`



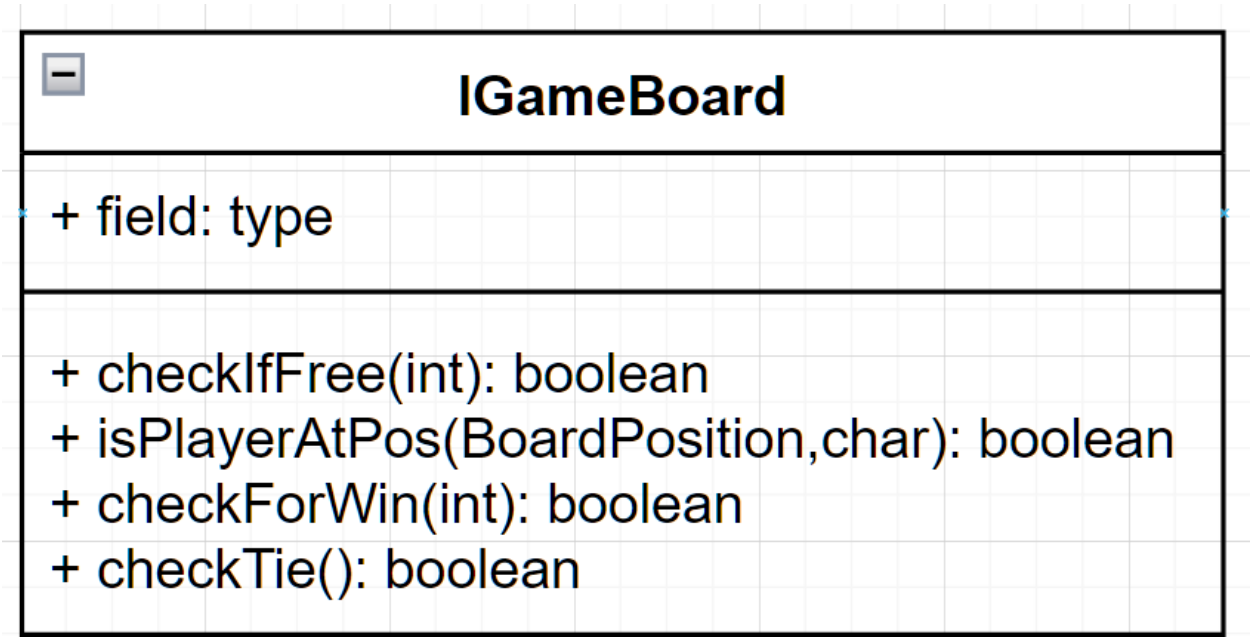
Class 3.1: `AbsGameBoard:`



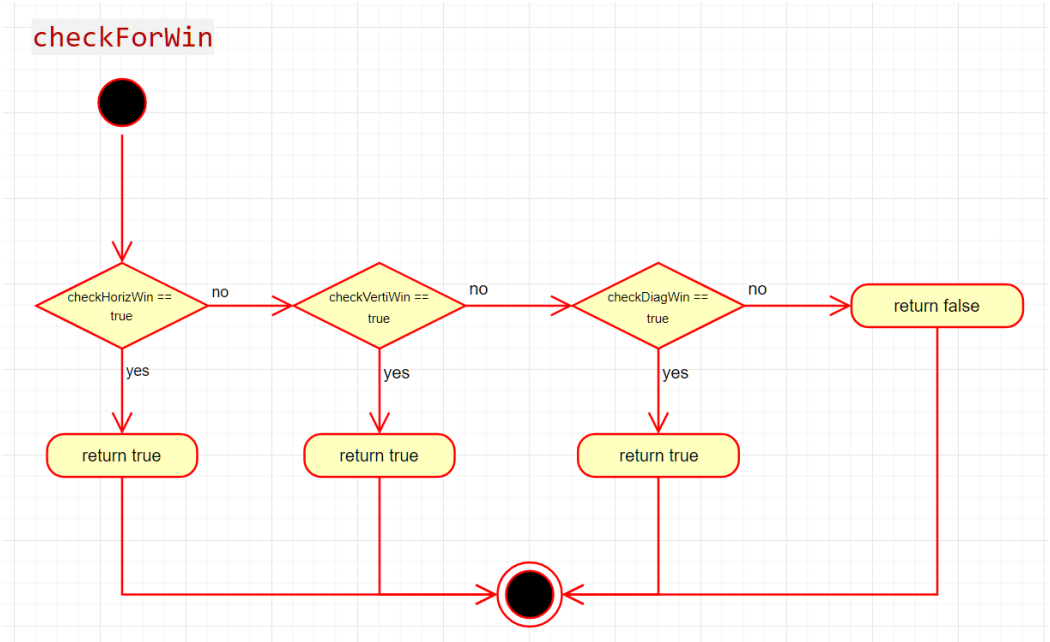
`toString`



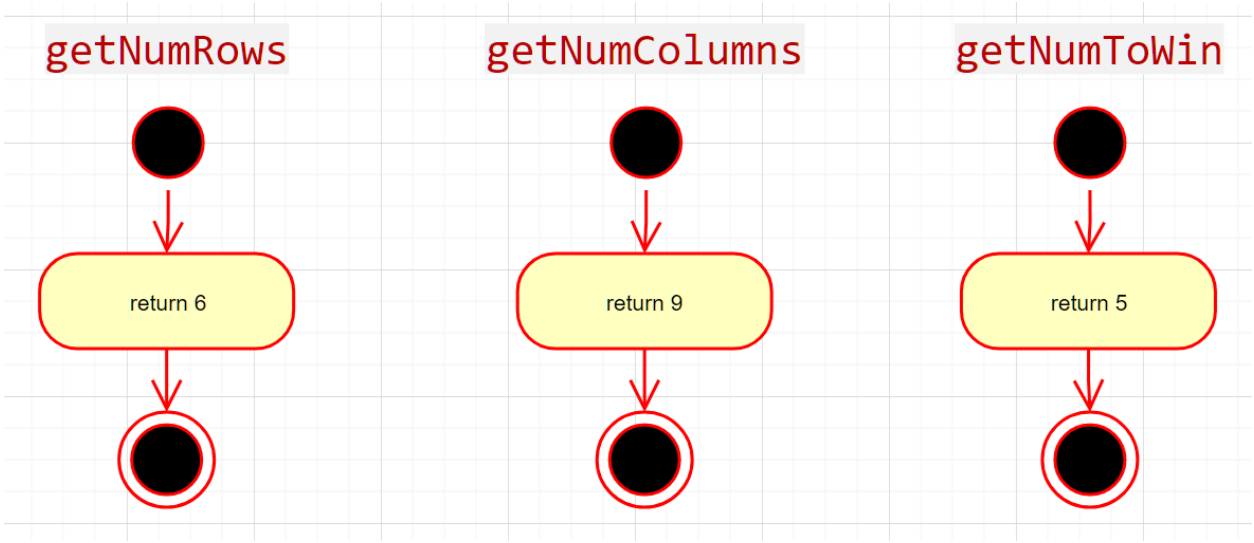
Class 3.2: `IGameBoard:`



`checkForWin`



getNumRows getNumColumns getNumToWin



Test Cases

Details in Project 4.