Julio Reyes

Professor Murali Sitaraman

CPSC 2150

February 7th 2021

Project Report

Requirements Analysis

Functional Requirements

1) As a user, I can view the current board, so that I know how the game is going.
2) As a user, I can enter a column position, so that I can place my piece in a specific position.
3) As a user, I can view the results of the game, so that I know who has won.
4) As a user, I can select how many rows I want my gameboard to have
5) As a user, I can select how many columns I want my gameboard to have
6) As a user, I can select how many players I want to play in the game
7) As a user, I can select my own unique player token
8) As a user, I can select whether I want a fast vs memory efficient game
9) As a user, I can be notified when I try to place my piece in an already filled column, so that I know an illegal move has been made.

10) As a user, I can have the option to play again after finishing a game, so that I can start a new game if I want to.
11) As a user, I can have the option to end the program after the game has finished, so that I can stop playing if I want to.
12) As a user, I can see when the game has ended in a tie, so that I know that no one has won.
13) As a user, I can see when a player has won due to placing five tokens in a row horizontally
14) As a user, I can see when a player has won due to placing five tokens in a row vertically
15) As a user, I can see when a player has won due to placing five tokens in a row diagonally
16) As a user, I can be given the option to place my token after my opponent's turn
17) As a user, I can be notified whenever a column I've chosen is already full.
18) As a user, I can be notified whenever I make a selection that is out of the bounds of the game board.
19) As a user, I can be notified if I choose a token that has already been chosen by another player

<u>Non Functional Requirements</u>

1) Must run on the Clemson School of Computing server.
2) Must be in Java.
3) Need to create UML class diagrams.
4) Need to create UML activity diagrams.

5) Need to create contracts for each method in my classes.
6) Create javadoc comments, specifying parameters, invariants, etc.
7) Game Board must be of user specified size
8) The bottom left of the board has coordinates [0, 0] and the top right of the board has coordinates [5, 8], depending on how many rows and columns the user wants to have (in this case it would be 6 rows and 9 columns.

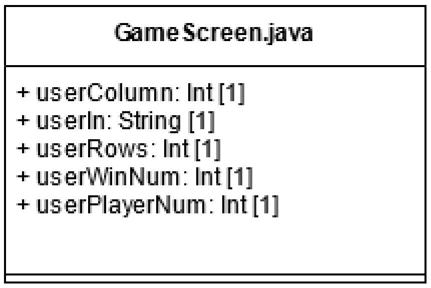
Deployment

Use "make" to compile all of the provided files. "make run" to run the actual program. "Make clean" can be used after running to remove any compiled class files. For our tests, "make test" will compile all of the test cases, while "make testGB" will run the tests for GameBoard, and "make testGBMem" will run the tests for GameBoardMem

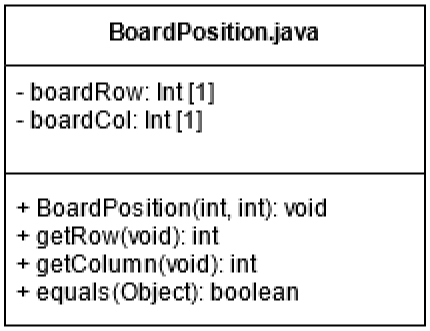
Design

UML Class Diagrams

`GameScreen.java`



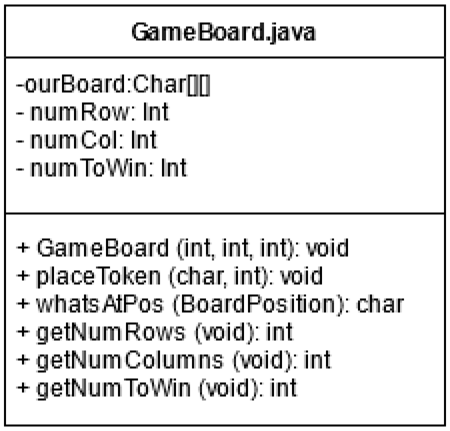
`BoardPosition.java`


BoardPosition.java

- boardRow: Int [1]
- boardCol: Int [1]

+ BoardPosition(int, int): void
+ getRow(void): int
+ getColumn(void): int
+ equals(Object): boolean

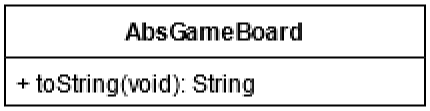
`GameBoard.java`


GameBoard.java

-ourBoard:Char[][]
- numRow: Int
- numCol: Int
- numToWin: Int

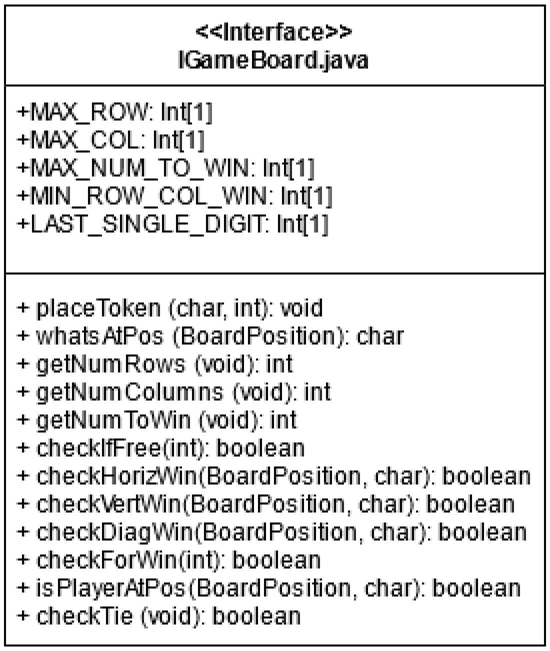
+ GameBoard (int, int, int): void
+ placeToken (char, int): void
+ whatsAtPos (BoardPosition): char
+ getNumRows (void): int
+ getNumColumns (void): int
+ getNumToWin (void): int

`AbsGameBoard.java`


AbsGameBoard

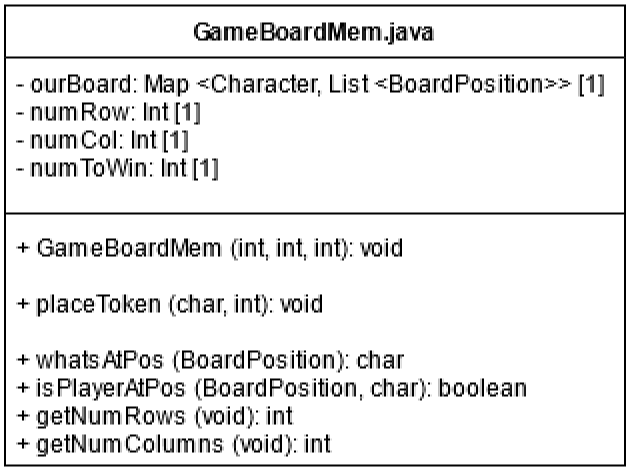
+ toString(void): String

IGameBoard.java



```
<<Interface>>
IGameBoard.java

+MAX_ROW: Int[1]
+MAX_COL: Int[1]
+MAX_NUM_TO_WIN: Int[1]
+MIN_ROW_COL_WIN: Int[1]
+LAST_SINGLE_DIGIT: Int[1]

+ placeToken (char, int): void
+ whatsAtPos (BoardPosition): char
+ getNumRows (void): int
+ getNumColumns (void): int
+ getNumToWin (void): int
+ checkIfFree(int): boolean
+ checkHorizWin(BoardPosition, char): boolean
+ checkVertWin(BoardPosition, char): boolean
+ checkDiagWin(BoardPosition, char): boolean
+ checkForWin(int): boolean
+ isPlayerAtPos(BoardPosition, char): boolean
+ checkTie (void): boolean
```

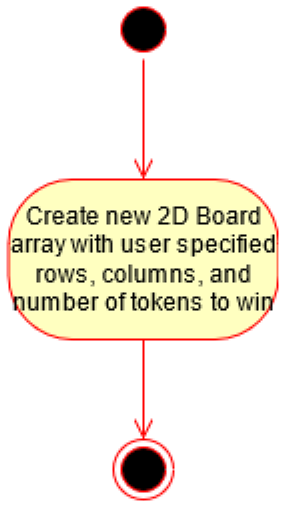
GameBoardMem.java



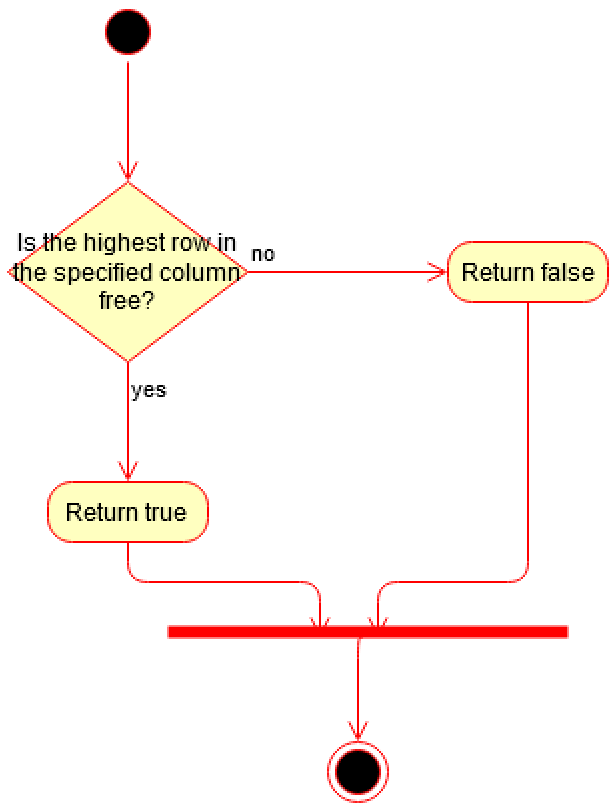
```
GameBoardMem.java

- ourBoard: Map <Character, List <BoardPosition>> [1]
- numRow: Int [1]
- numCol: Int [1]
- numToWin: Int [1]

+ GameBoardMem (int, int, int): void

+ placeToken (char, int): void

+ whatsAtPos (BoardPosition): char
+ isPlayerAtPos (BoardPosition, char): boolean
+ getNumRows (void): int
+ getNumColumns (void): int
```

UML Activity Diagrams

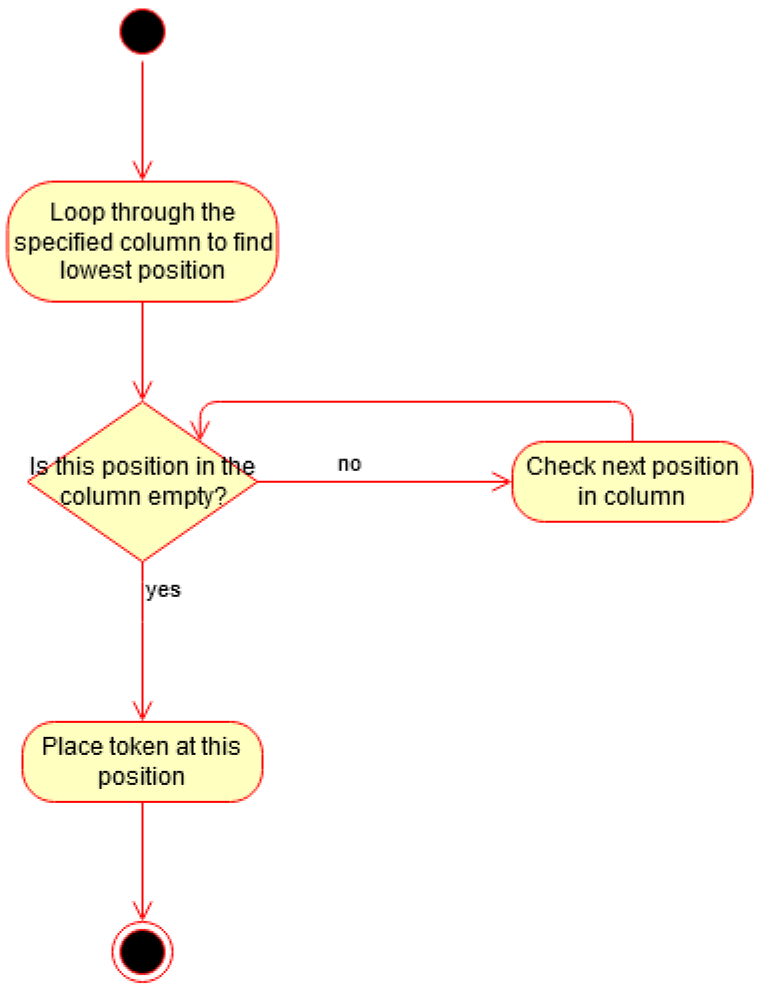
`GameBoard.java - GameBoard()`


Create new 2D Board array with user specified rows, columns, and number of tokens to win

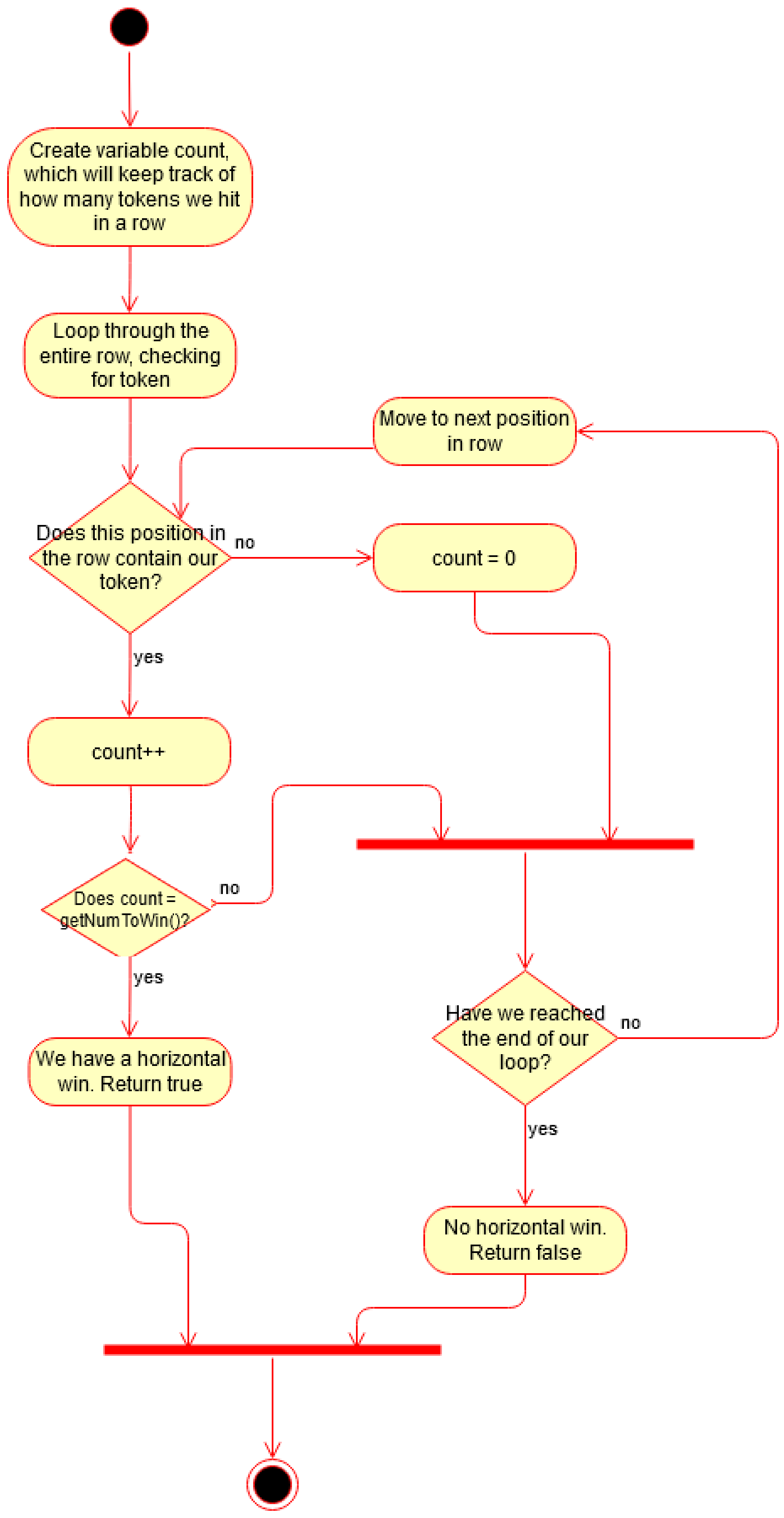
IGameBoard - checkIfFree()



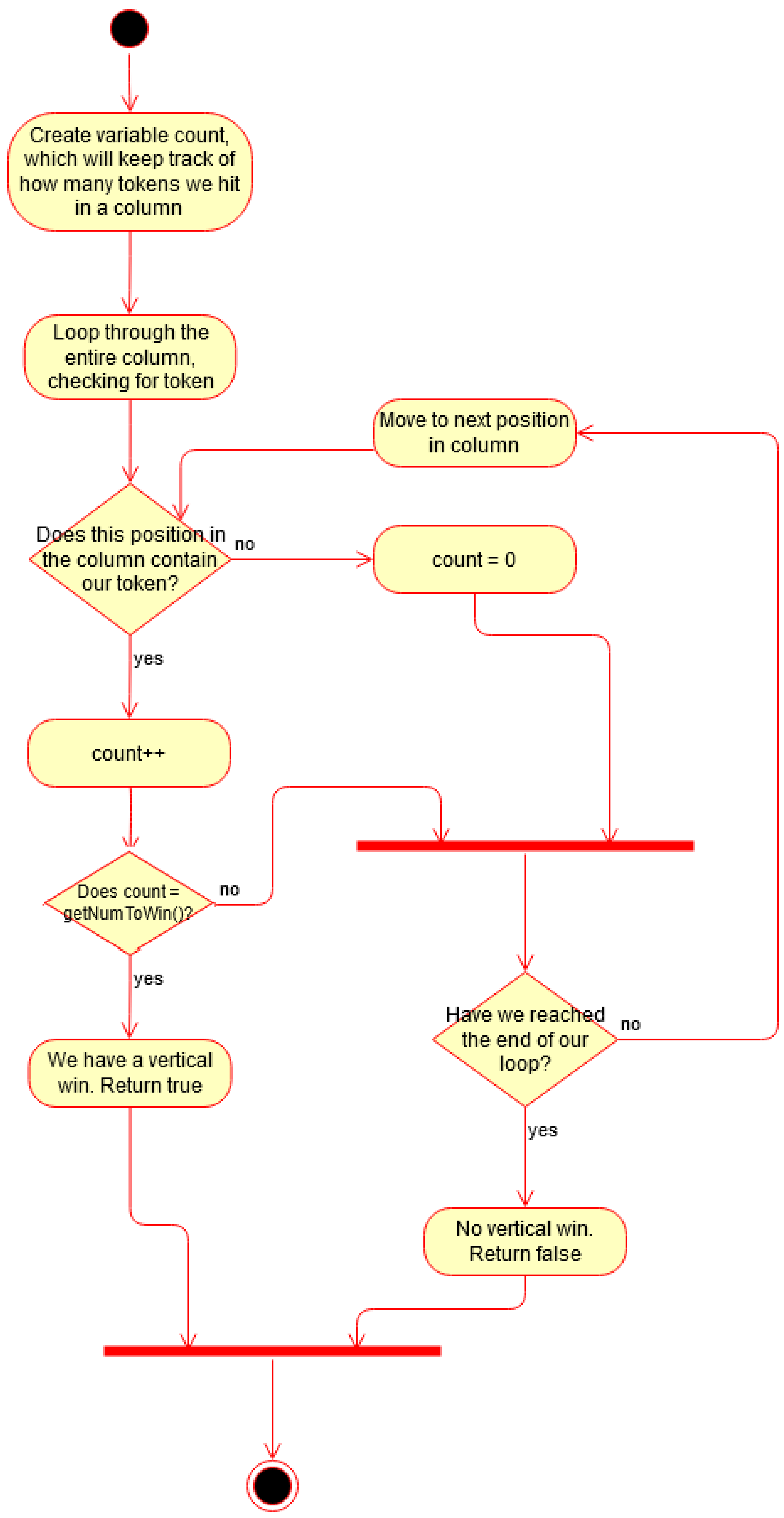
GameBoard - placeToken()


Loop through the specified column to find lowest position
Is this position in the column empty?
no
Check next position in column
yes
Place token at this position

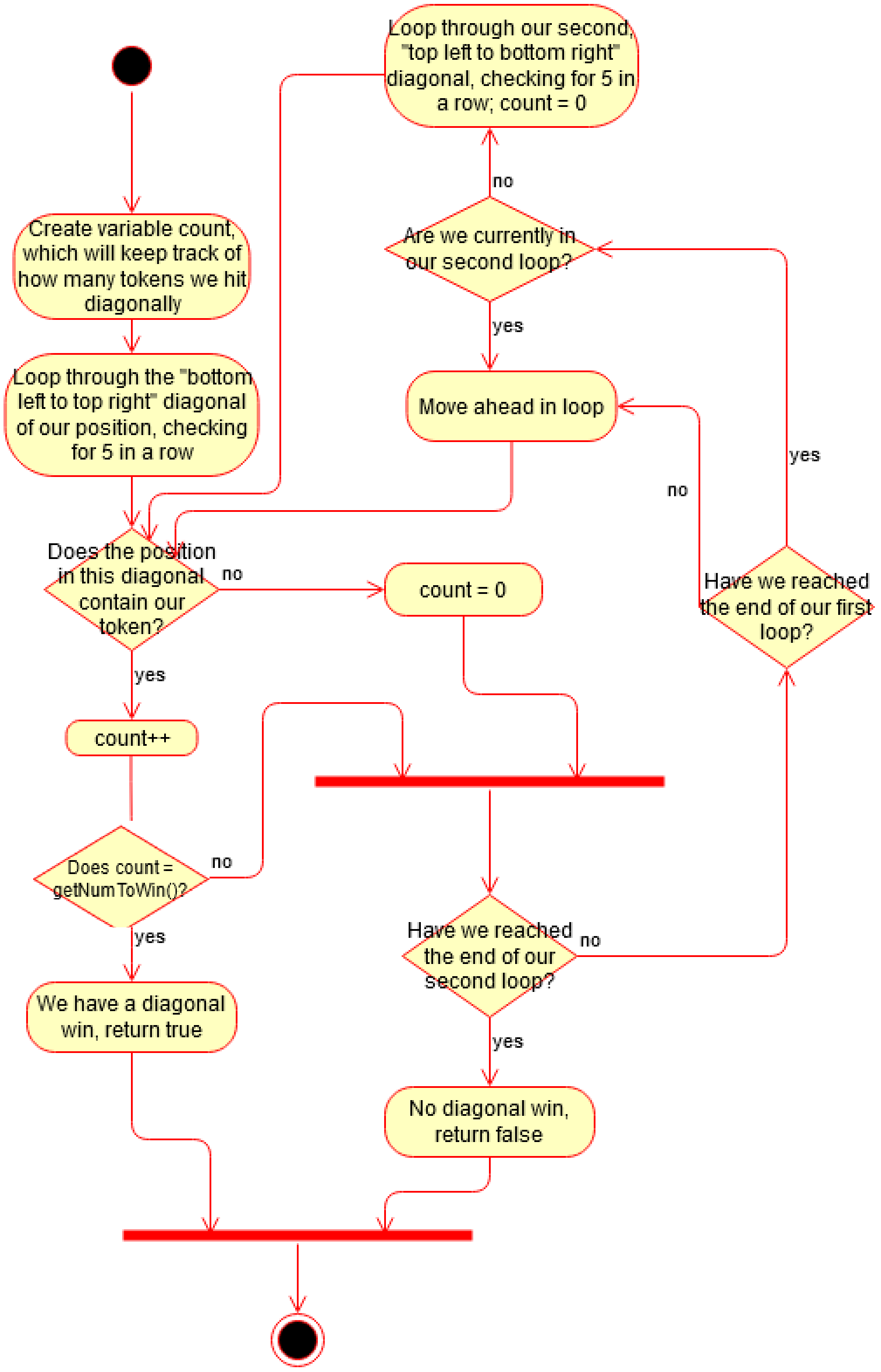
IGameBoard - checkHorizWin()


Create variable count, which will keep track of how many tokens we hit in a row
Loop through the entire row, checking for token
Move to next position in row
Does this position in the row contain our token?
no
count = 0
yes
count++
Does count = getNumToWin()?
no
yes
We have a horizontal win. Return true
Have we reached the end of our loop?
no
yes
No horizontal win. Return false

IGameBoard - checkVertWin()

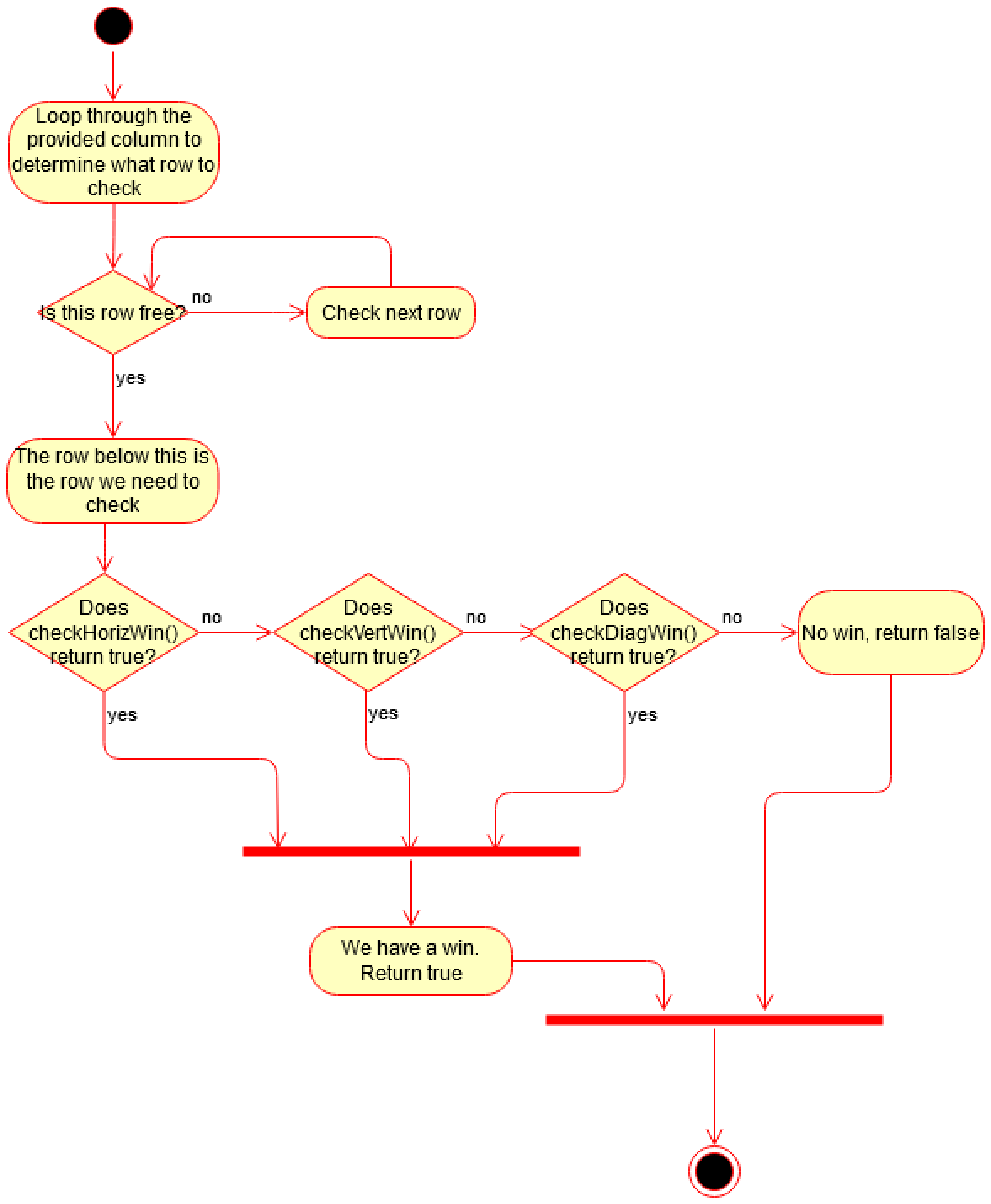

Create variable count, which will keep track of how many tokens we hit in a column
Loop through the entire column, checking for token
Move to next position in column
Does this position in the column contain our token?
no
count = 0
yes
count++
Does count = getNumToWin()?
no
yes
We have a vertical win. Return true
Have we reached the end of our loop?
no
yes
No vertical win. Return false

`IGameBoard - checkDiagWin()`

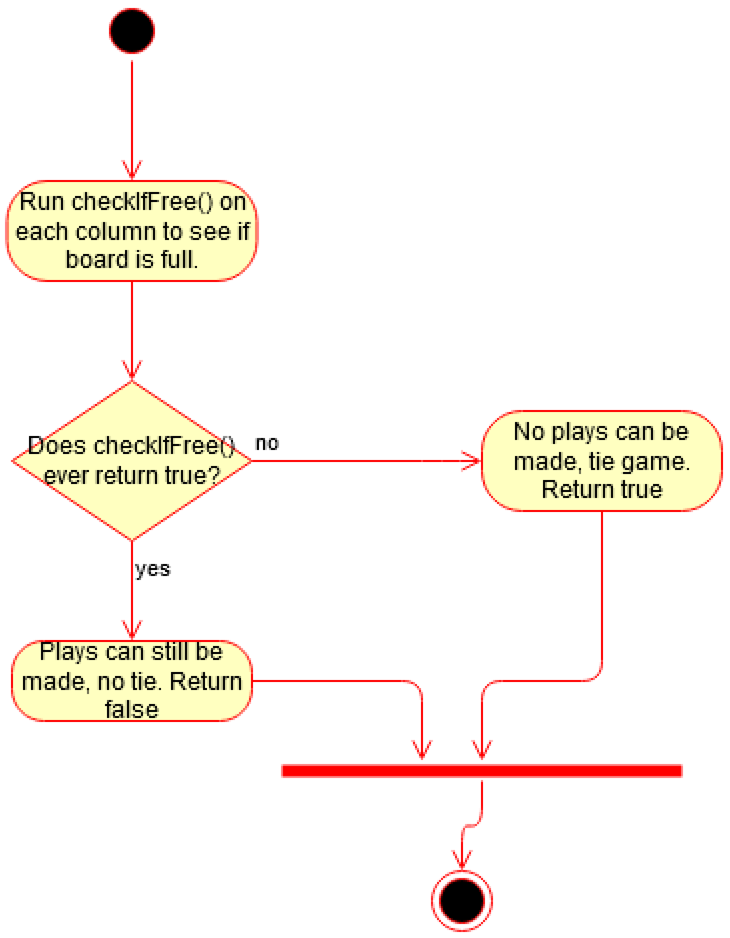


Create variable count, which will keep track of how many tokens we hit diagonally
Loop through the "bottom left to top right" diagonal of our position, checking for 5 in a row
Loop through our second "top left to bottom right" diagonal, checking for 5 in a row; count = 0
no
Are we currently in our second loop?
yes
Move ahead in loop
yes
no
Does the position in this diagonal contain our token?
no
count = 0
Have we reached the end of our first loop?
yes
count++
Does count = getNumToWin()?
no
yes
We have a diagonal win, return true
Have we reached the end of our second loop?
no
yes
No diagonal win, return false

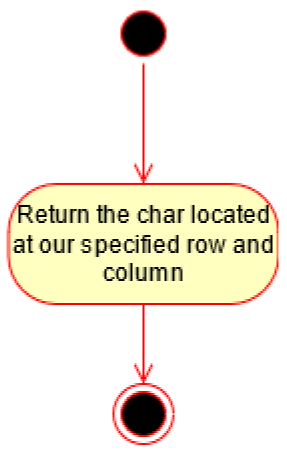
IGameBoard - checkForWin()


Loop through the provided column to determine what row to check
Is this row free?
no
Check next row
yes
The row below this is the row we need to check
Does checkHorizWin() return true?
no
Does checkVertWin() return true?
no
Does checkDiagWin() return true?
no
No win, return false
yes
yes
yes
We have a win. Return true

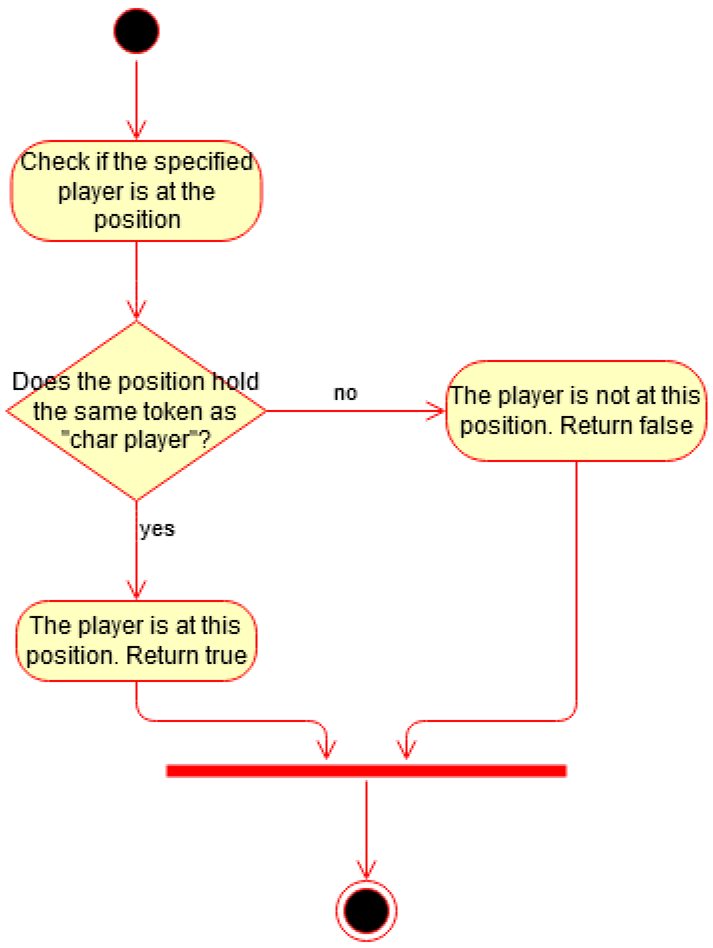
IGameboard - checkTie()


Run checkIfFree() on each column to see if board is full.
Does checkIfFree() ever return true?
no
No plays can be made, tie game. Return true
yes
Plays can still be made, no tie. Return false

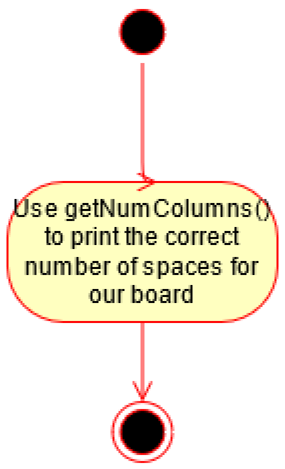
`Gameboard - whatsAtPos()`



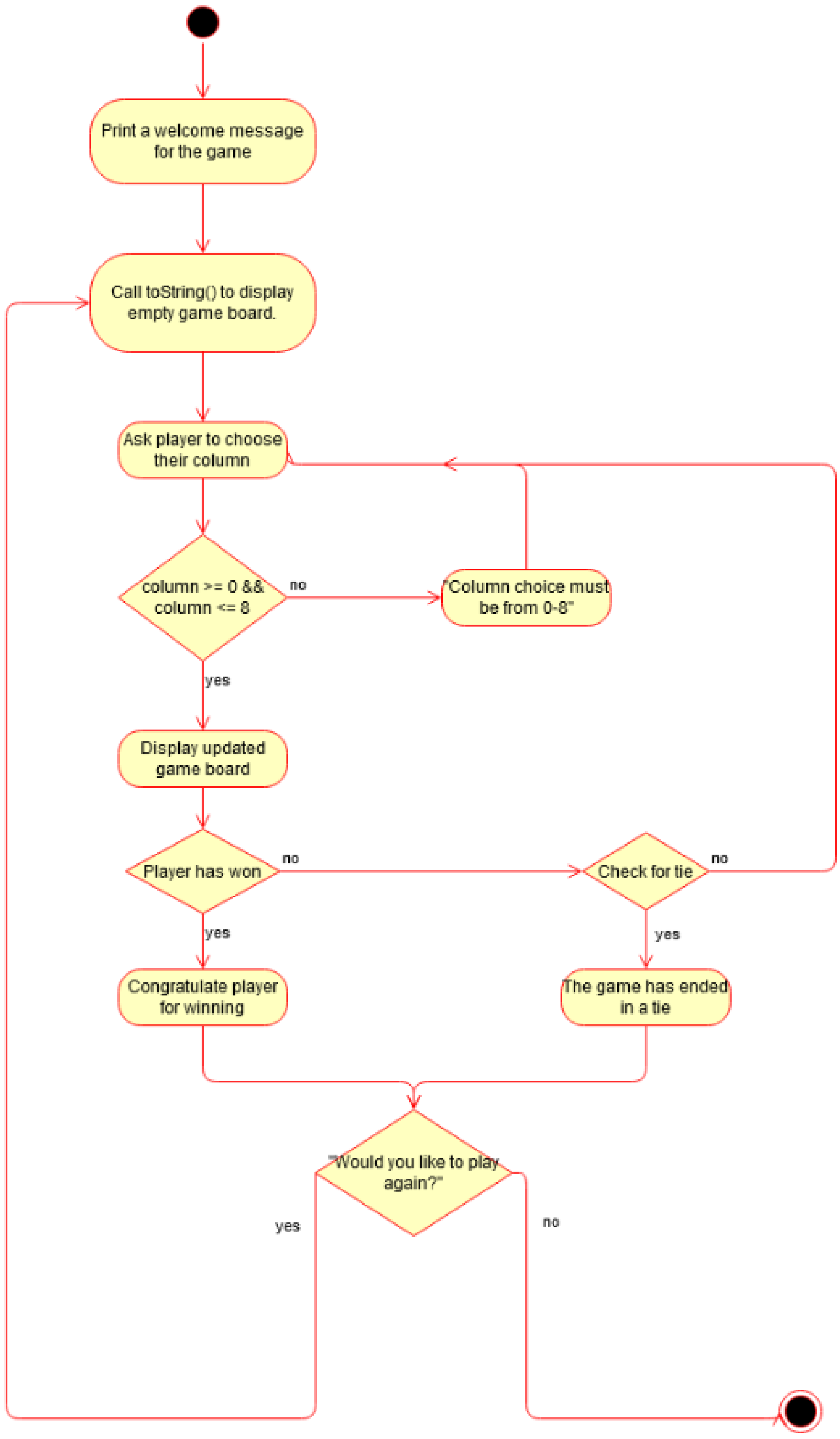
`IGameBoard - isPlayerAtPos()`


Check if the specified player is at the position
Does the position hold the same token as "char player"?
no
The player is not at this position. Return false
yes
The player is at this position. Return true

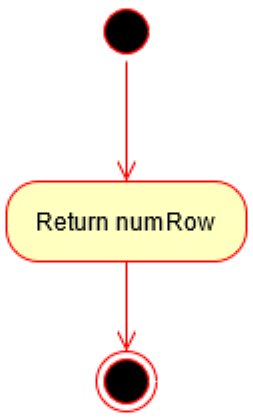
AbsGameBoard - toString()


Use getNumColumns() to print the correct number of spaces for our board

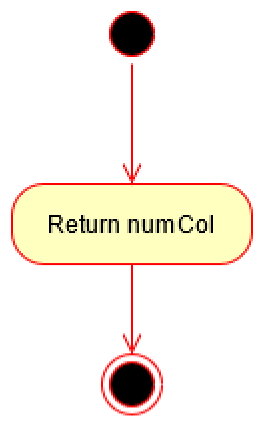
GameScreen.java - main()


Print a welcome message for the game
Call toString() to display empty game board.
Ask player to choose their column
column >= 0 && column <= 8
no
"Column choice must be from 0-8"
yes
Display updated game board
Player has won
no
Check for tie
no
yes
Congratulate player for winning
yes
The game has ended in a tie
"Would you like to play again?"
yes
no

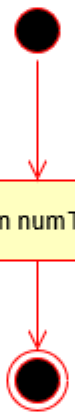
`GameBoard.java AND GameBoardMem.java - getNumRows()`



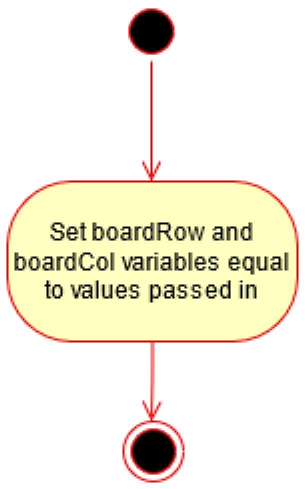
GameBoard.java AND GameBoardMem.java - getNumColumns()



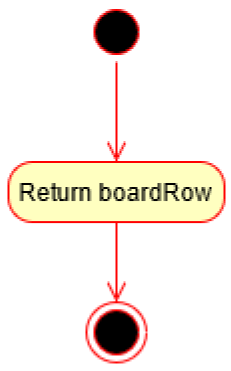
`GameBoard.java - getNumToWin()`



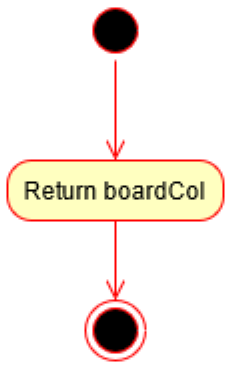
BoardPosition.java - BoardPosition()


Set boardRow and boardCol variables equal to values passed in

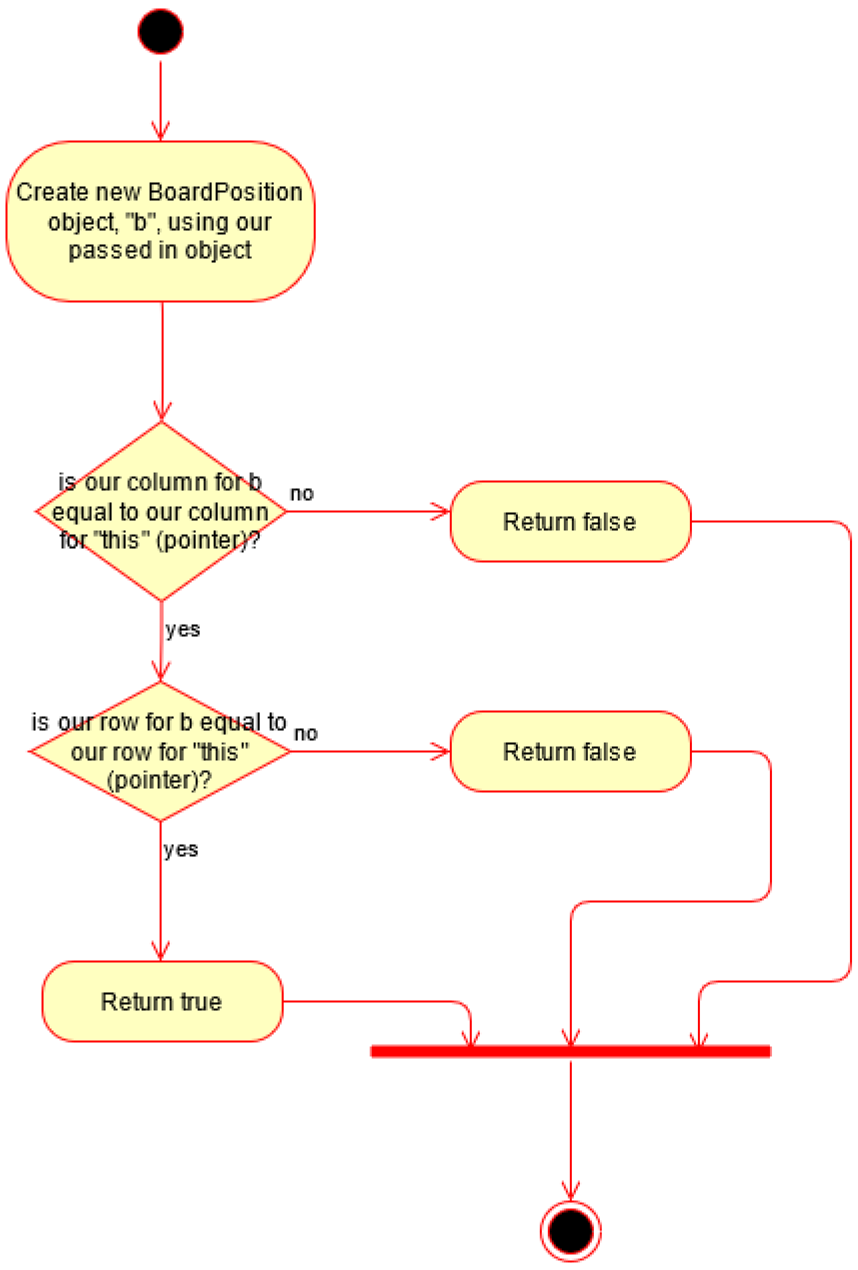
`BoardPosition.java - getRow()`



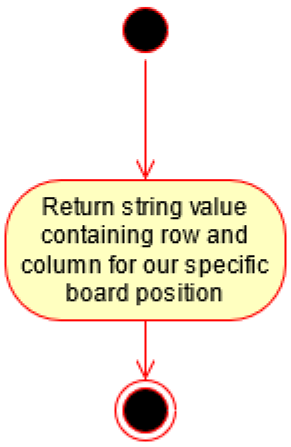
`BoardPosition.java - getColumn()`



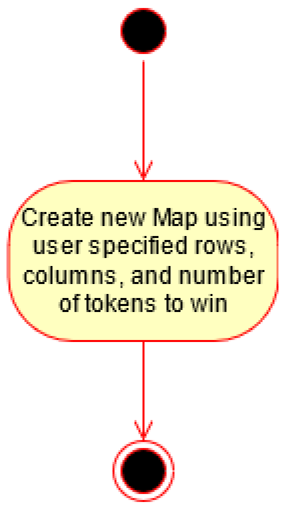
`BoardPosition.java - equals()`


Create new BoardPosition object, "b", using our passed in object
is our column for b equal to our column for "this" (pointer)?
no
Return false
yes
is our row for b equal to our row for "this" (pointer)?
no
Return false
yes
Return true

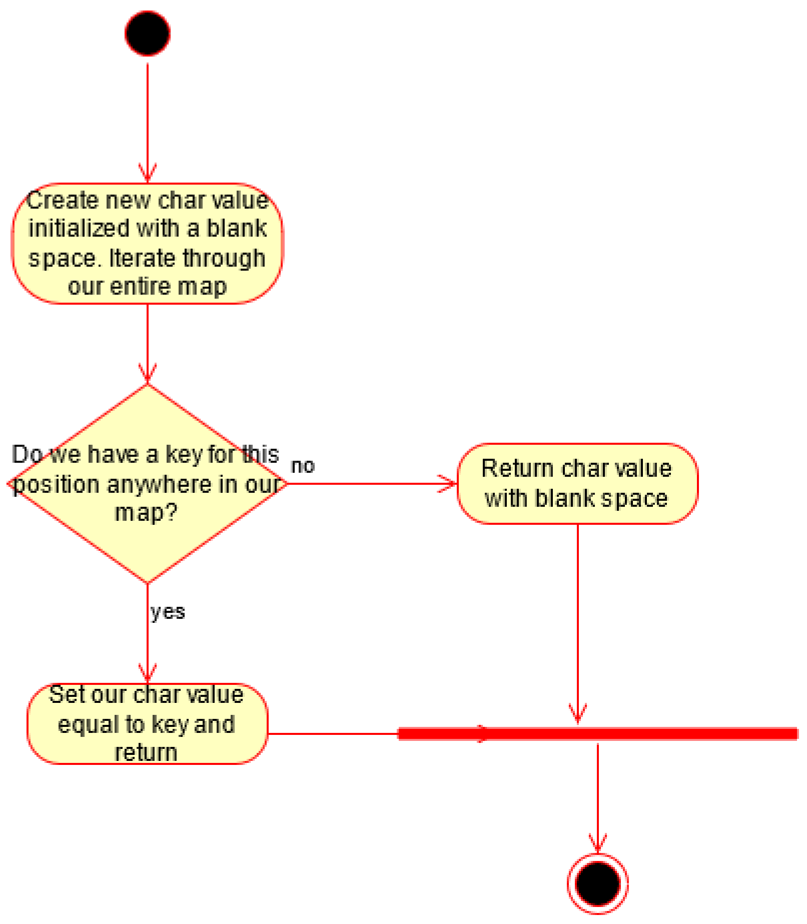
`BoardPosition.java - toString`


Return string value containing row and column for our specific board position

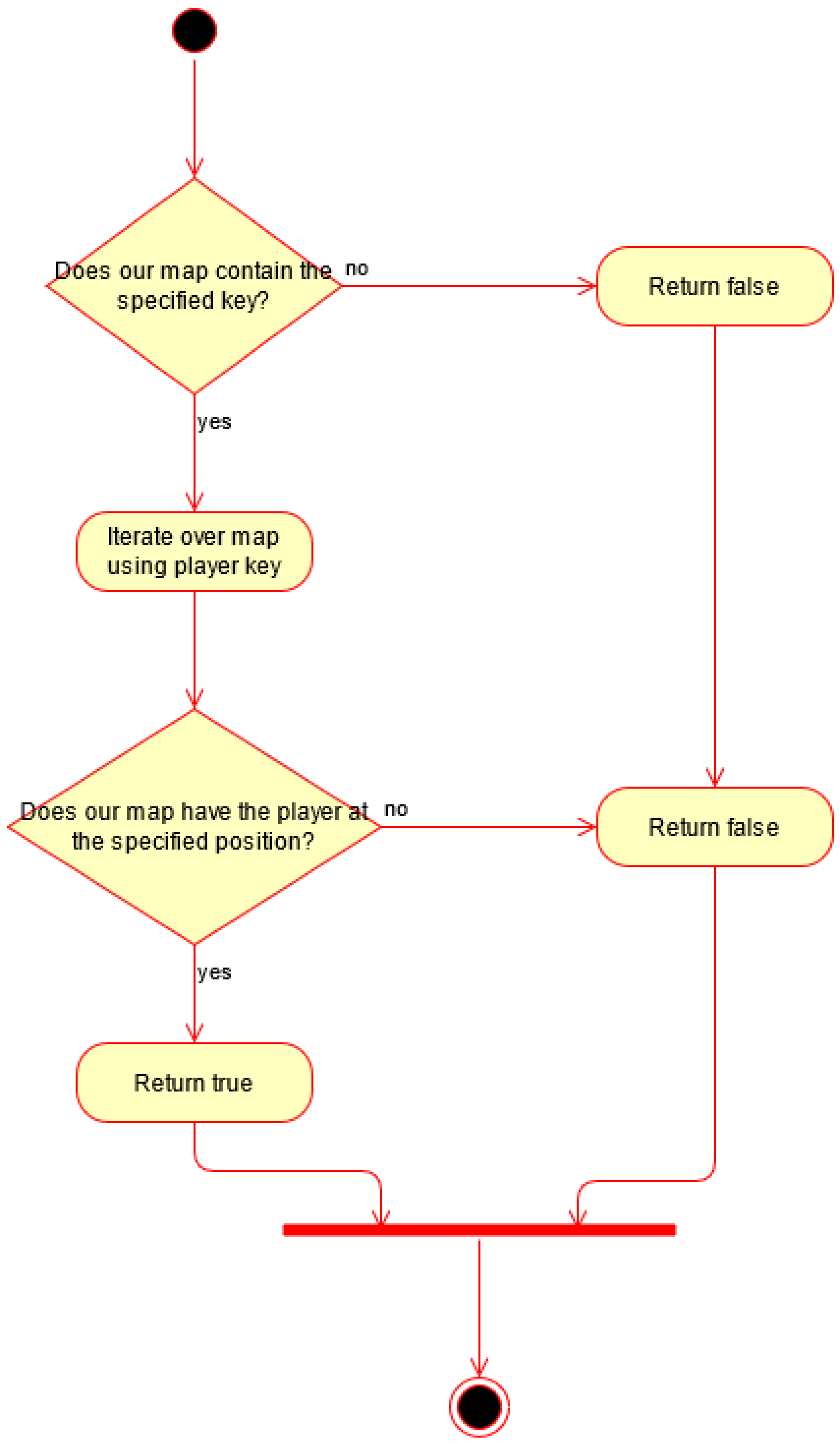
GameBoardMem.java - GameBoardMem()


Create new Map using user specified rows, columns, and number of tokens to win

`GameBoardMem.java - whatsAtPos()`


Create new char value initialized with a blank space. Iterate through our entire map
Do we have a key for this position anywhere in our map?
no
Return char value with blank space
yes
Set our char value equal to key and return

`GameBoardMem - isPlayerAtPos()`


Does our map contain the specified key?
no
Return false
yes
Iterate over map using player key
Does our map have the player at the specified position?
no
Return false
yes
Return true