

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 be notified when I try to place my piece in an already filled column, so that I know an illegal move has been made.
- 5) 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.
- 6) 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.
- 7) As a user, I can see when the game has ended in a tie, so that I know that no one has won.

- 8) As a user, I can see when a player has won due to placing five tokens in a row horizontally
- 9) As a user, I can see when a player has won due to placing five tokens in a row vertically
- 10) As a user, I can see when a player has won due to placing five tokens in a row diagonally
- 11) As a user, I can be given the option to place my token after my opponent's turn
- 12) As a user, I can be notified whenever a column I've chosen is already full.
- 13) As a user, I can be notified whenever I make a selection that is out of the bounds of the game board.

Non Functional Requirements

- 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 6 x 9 in size
- 8) The bottom left of the board has coordinates [0, 0] and the top right of the board has coordinates [5, 8]
- 9) Player X goes first

Design

UML Class Diagrams

GameScreen.java

GameScreen
+ Column: Int[1]
+ main(void): int

BoardPosition.java

BoardPosition
-Row: Int [1] -Column: Int [1]
+ BoardPosition(int, int): void + getRow(void): int + getColumn(void): int + equals(Object): bool + toString(void): String

GameBoard.java

GameBoard
- ourBoard: Char[5][8]
+ GameBoard(void): void + placeToken(char, int): void + whatsAtPos(BoardPosition): char

AbsGameBoard.java

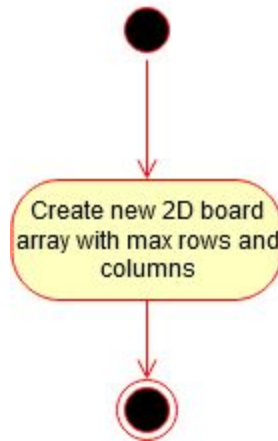
AbsGameBoard
+ toString(void): String

IGameBoard.java

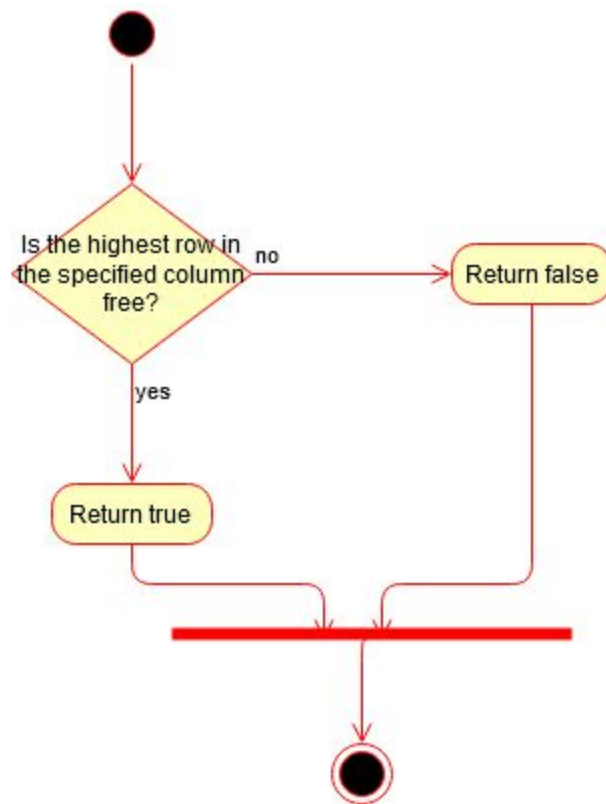
<<Interface>> IGameBoard
+MAX_ROW: Int [1] +MAX_COLUMN: Int [1] +NUM_TO_WIN: Int [1]
+ placeToken(char, int): void + whatsAtPos(BoardPosition): char + checkIfFree(int): bool + checkHorizWin(BoardPosition, char): bool + checkVertWin(BoardPosition, char): bool + checkDiagWin(BoardPosition, char): bool + checkForWin (int): bool + isPlayerAtPos (BoardPosition, char): bool + checkTie(void): bool + getNumRows(void): int + getNumColumns(void): int + getNumToWin(void): int

UML Activity Diagrams

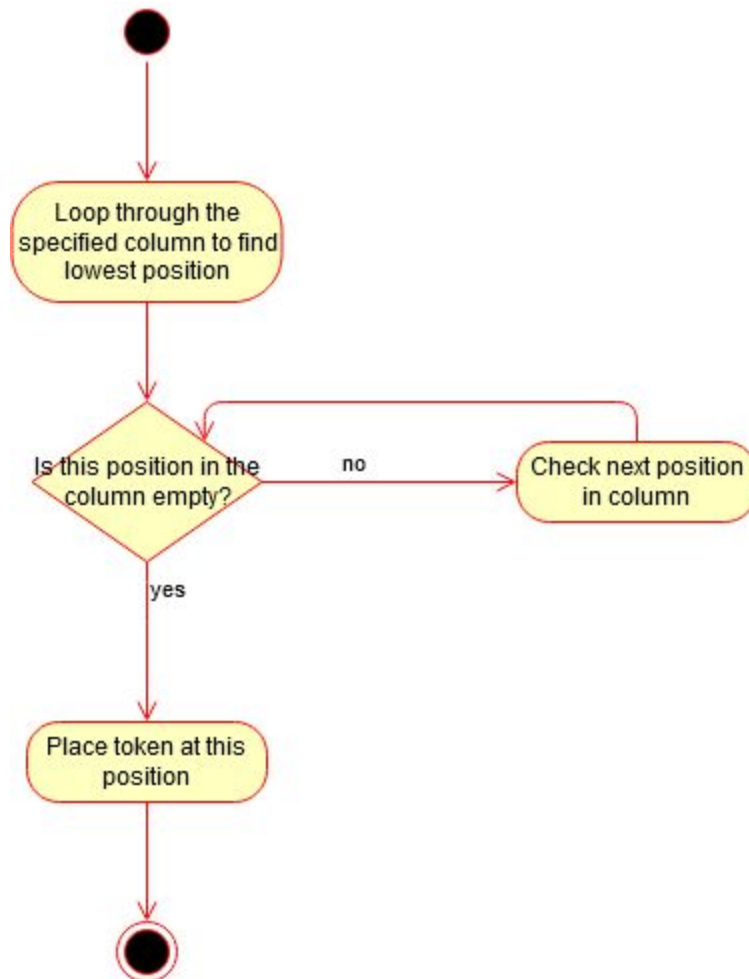
GameBoard.java - GameBoard()



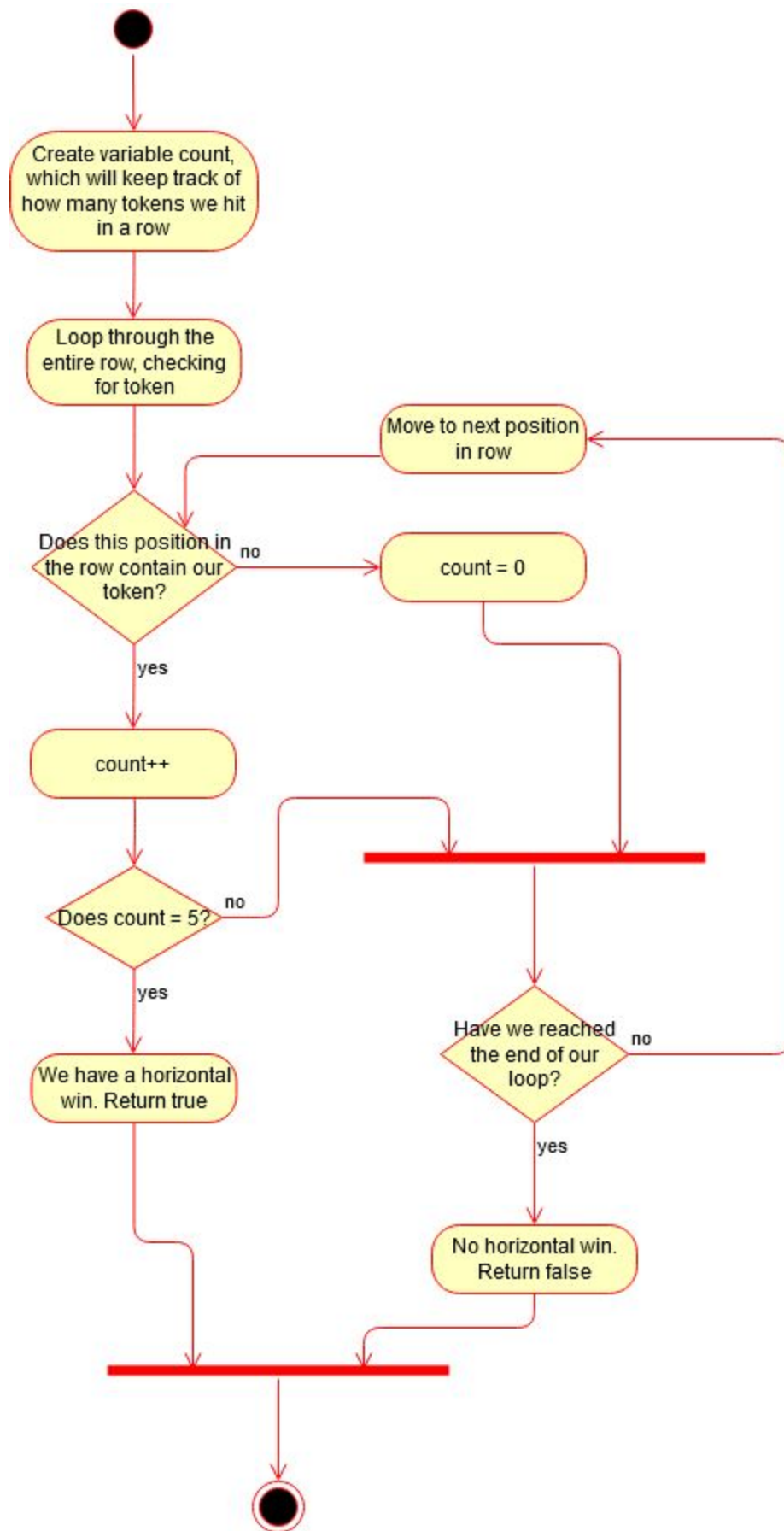
IGameBoard - checkIfFree()



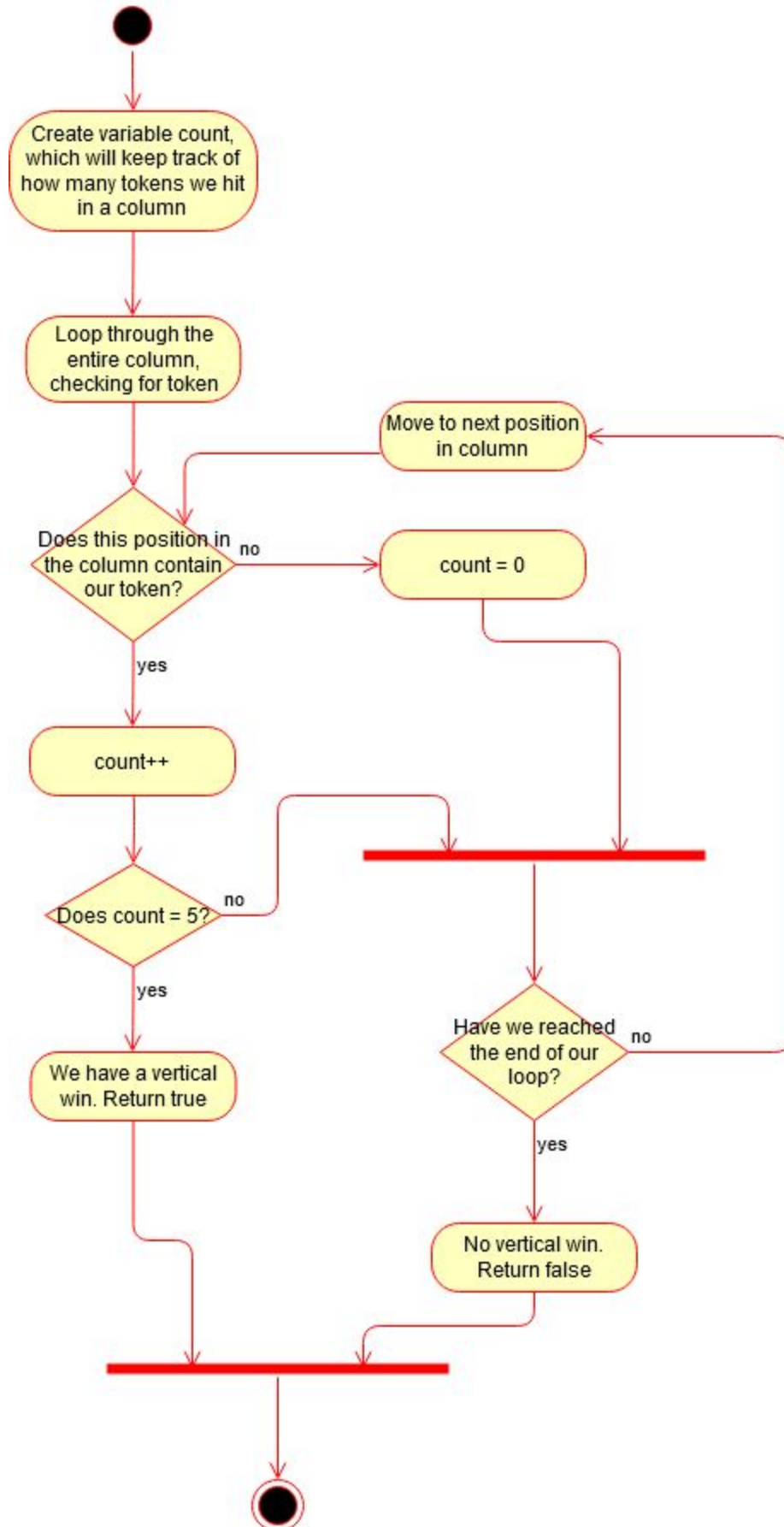
GameBoard - placeToken()



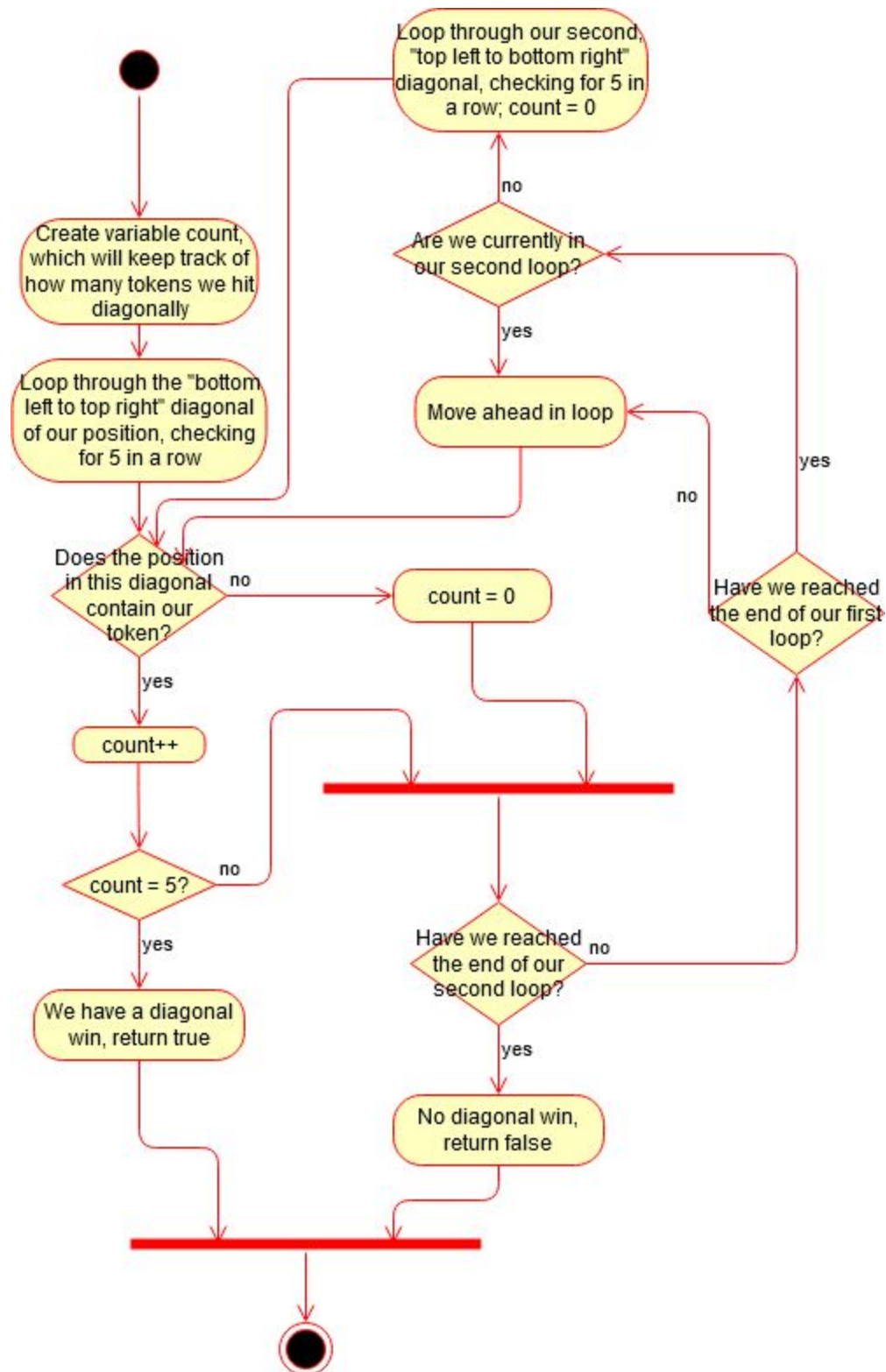
IGameBoard - checkHorizWin()



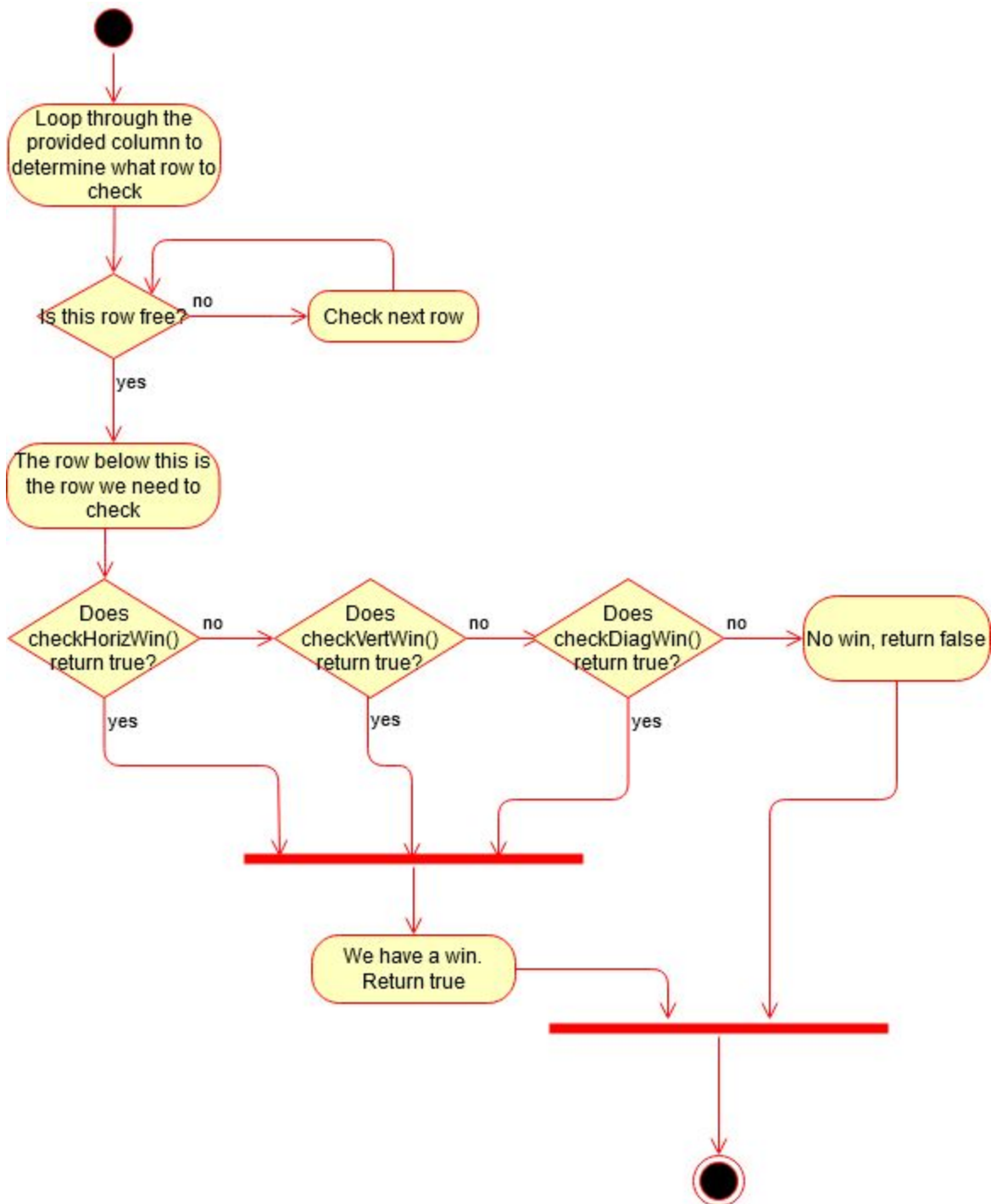
IGameBoard - checkVertWin()



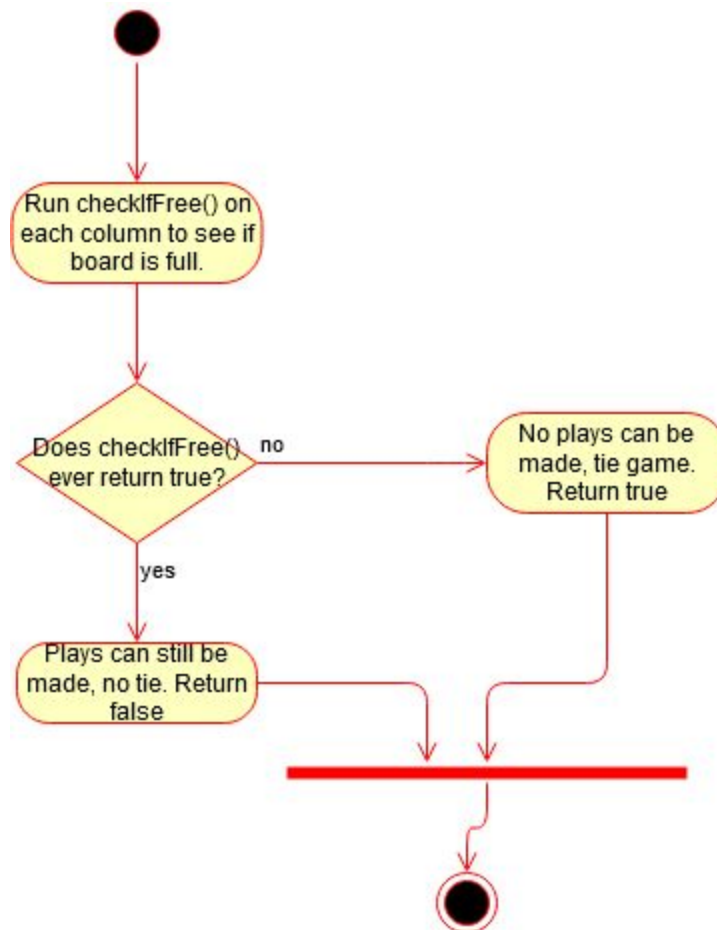
IGameBoard - checkDiagWin()



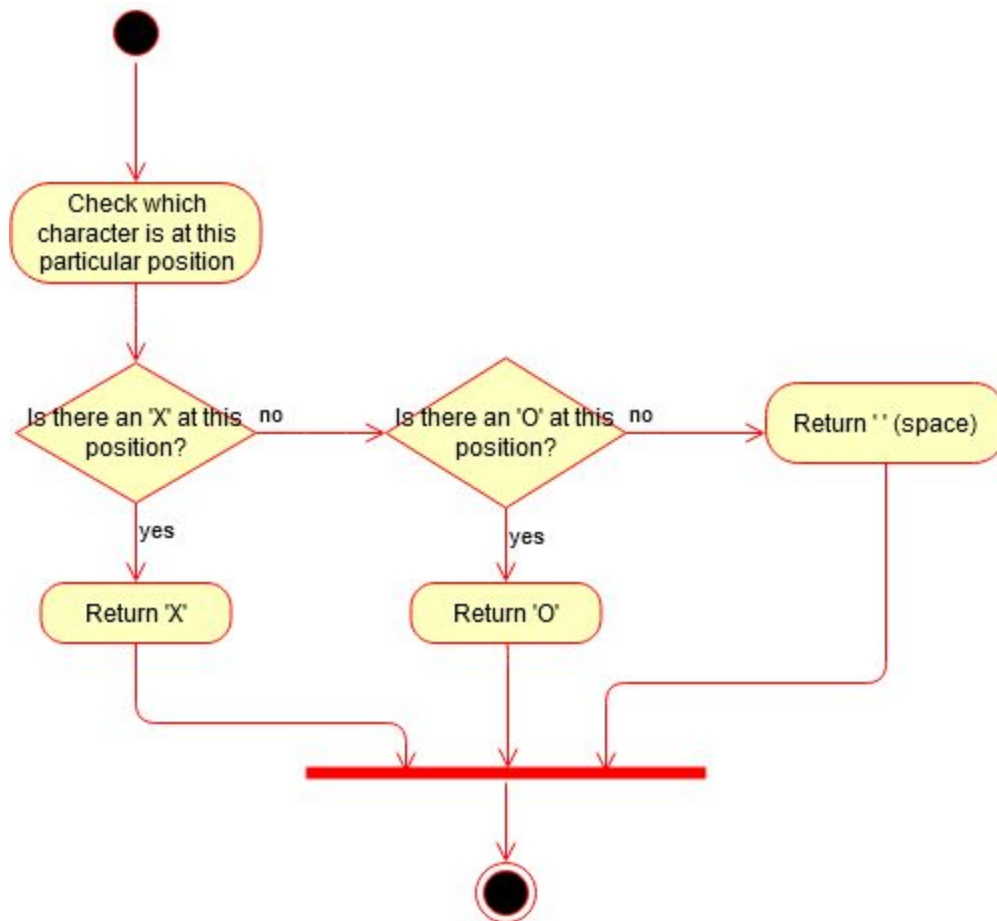
IGameBoard - checkForWin()



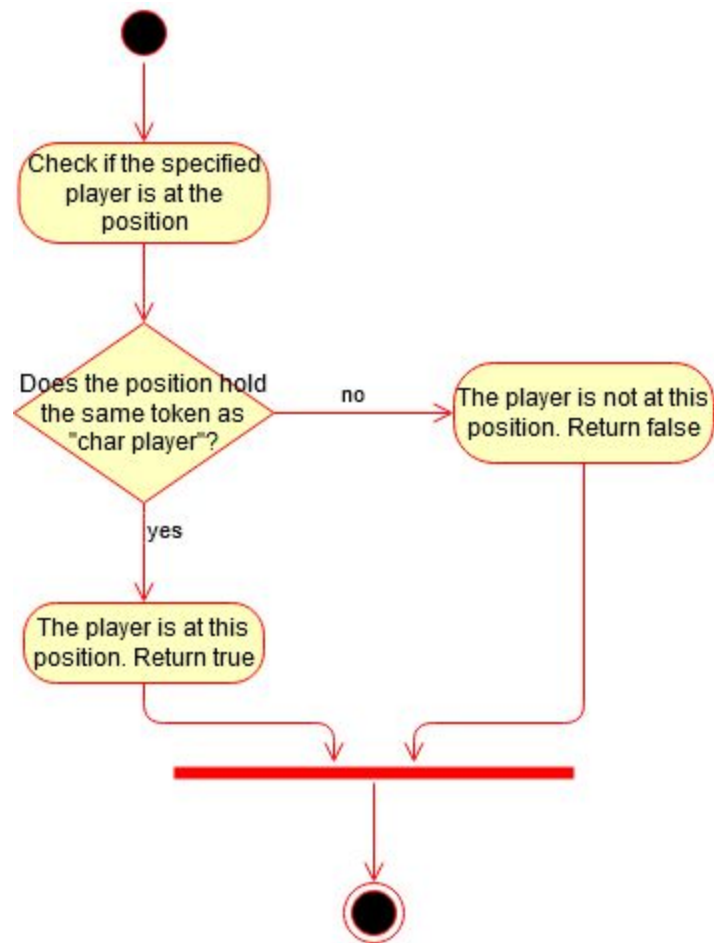
IGameboard - checkTie()



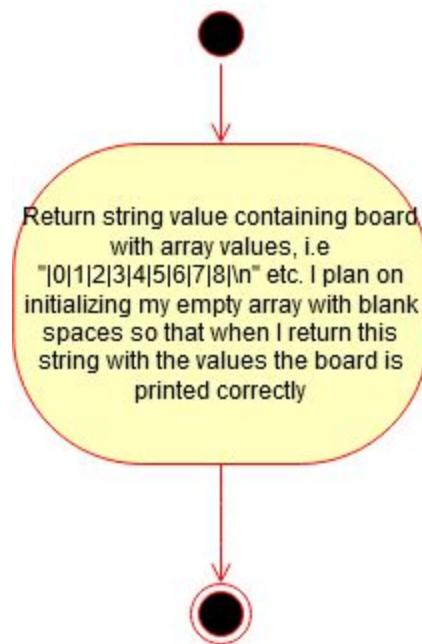
Gameboard - whatsAtPos()



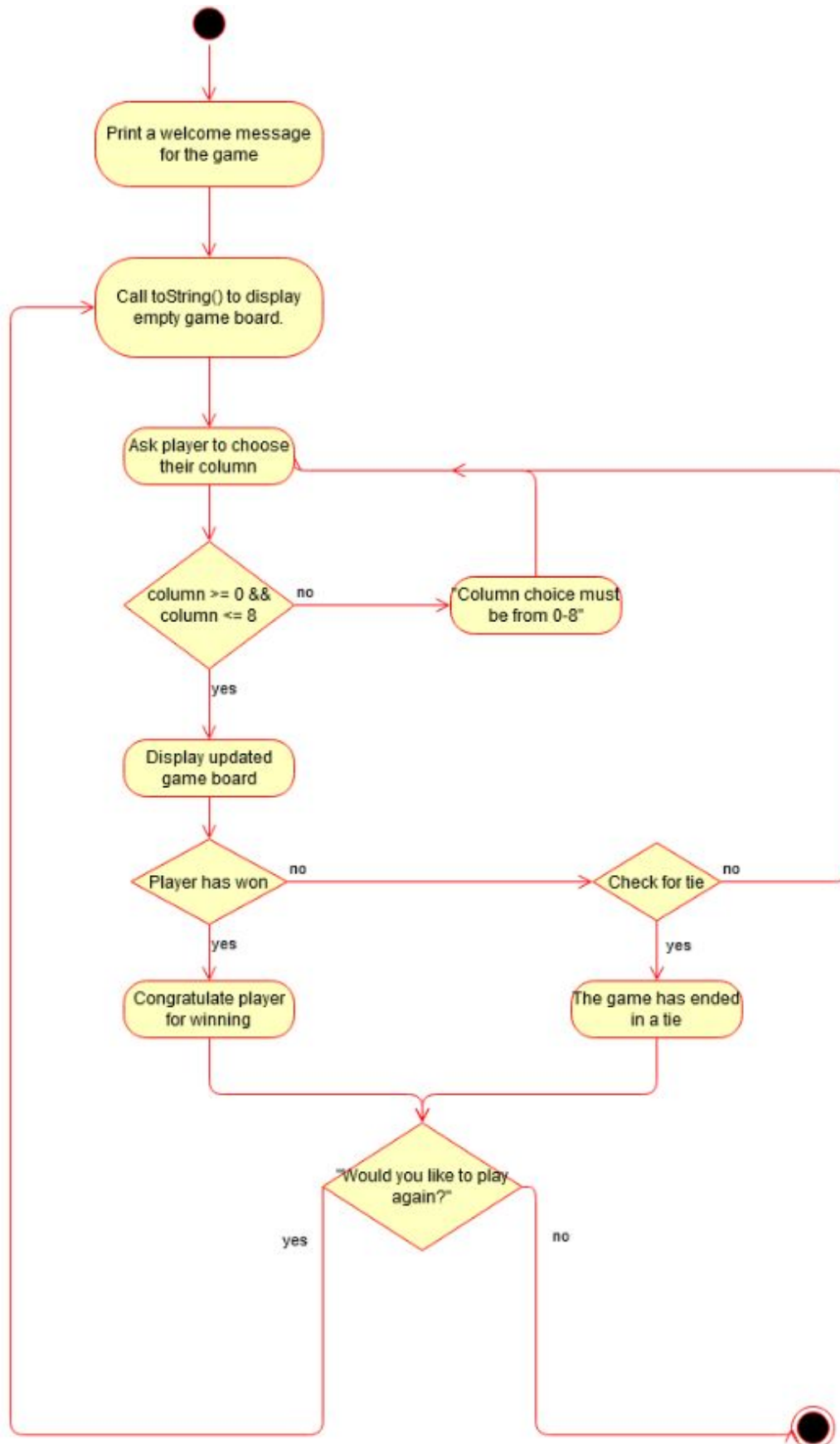
IGameBoard - isPlayerAtPos()



AbsGameBoard - toString()



GameScreen.java - main()



IGameBoard.java - getNumRows()



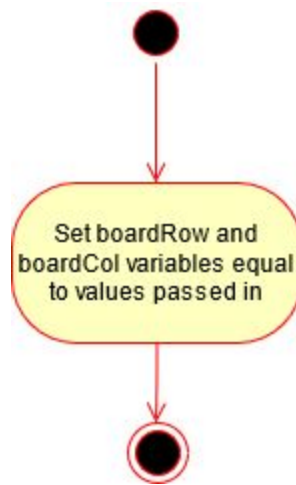
IGameBoard.java - getNumColumns()



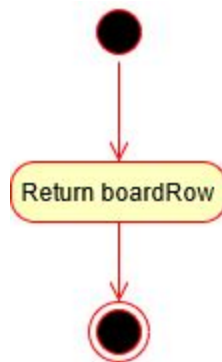
IGameBoard.java - getNumToWin()



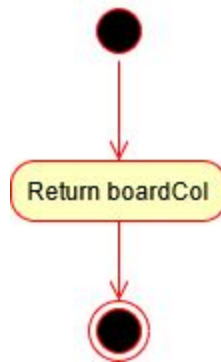
BoardPosition.java - BoardPosition()



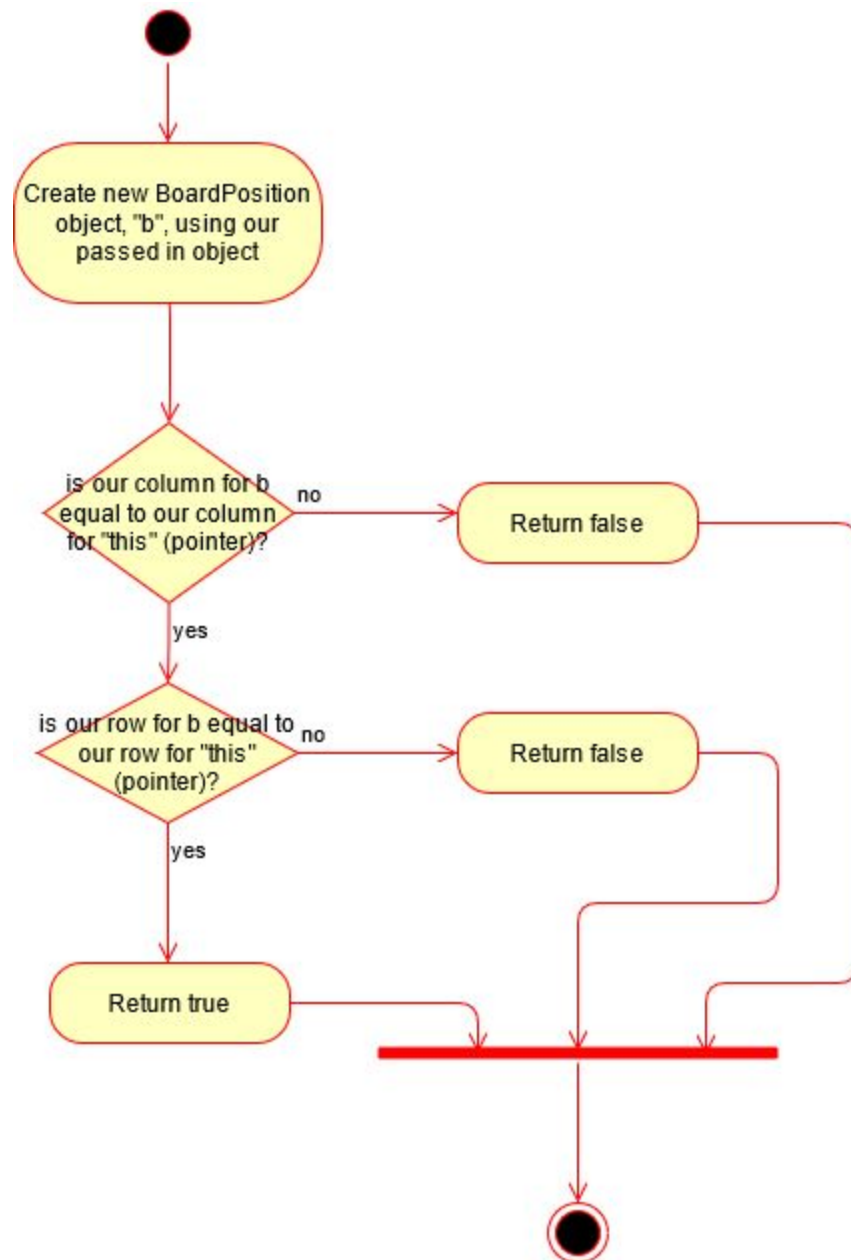
BoardPosition.java - getRow()



BoardPosition.java - getColumn()



BoardPosition.java - equals()



BoardPosition.java - toString

