Paige Clemons (pclemon)
Lab Section 5
Homework 2
Due: 10/10/18

Requirement Analysis

Functional Requirements

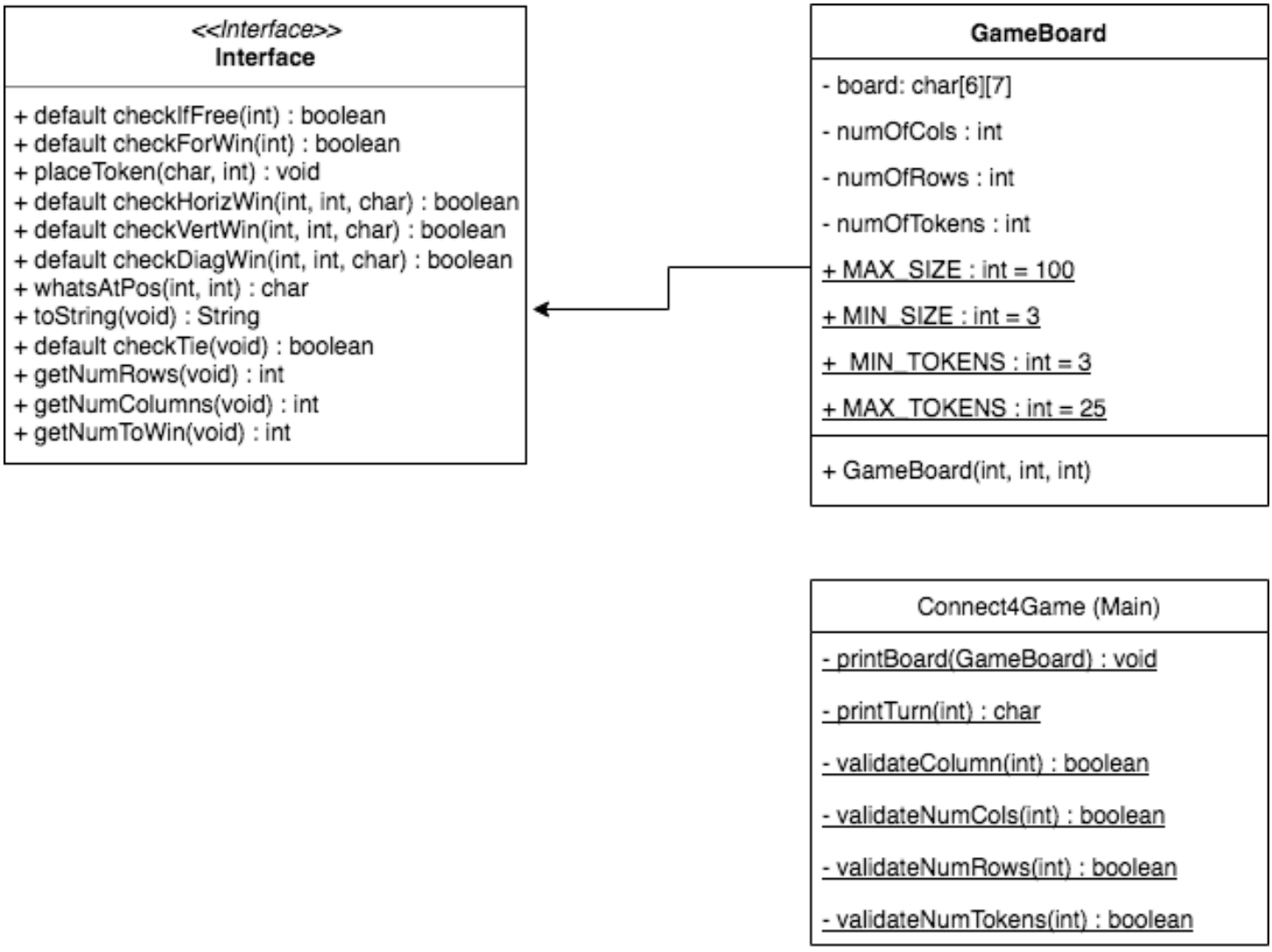
- Player should be able to see the board in order to know where they can place their tokens.
- Players should be able to choose where to place their tokens in order to effectively play the game to beat their opponent.
- Players should be able to choose the size of the board so that they can play the game on whatever size board they want.
- Players should be able to choose how many tokens in a row they need to win so that they can play the game in the way they choose.
- Players should be able to choose if they want to play again in order to have a rematch with an opponent.

Non-functional Requirements

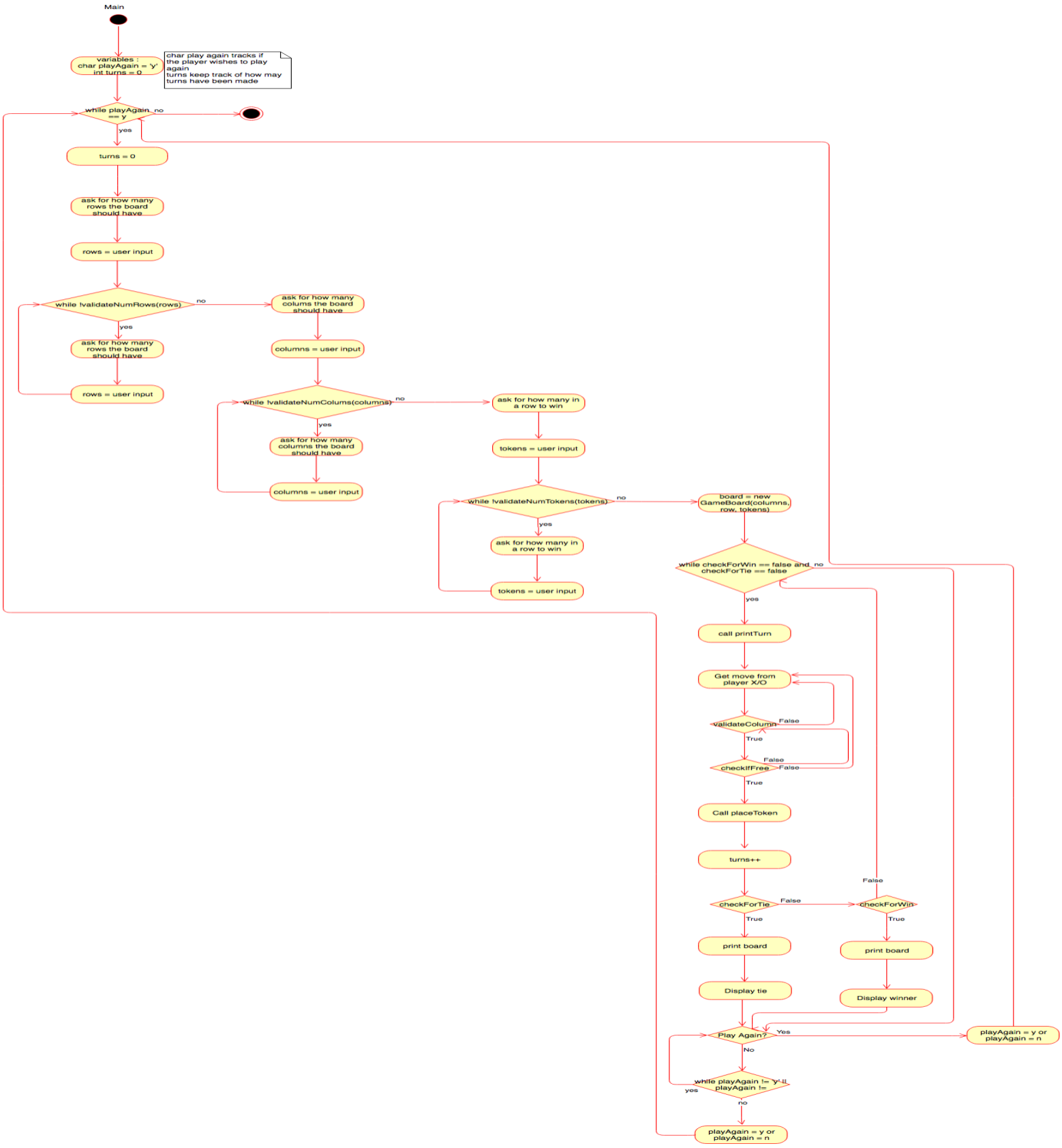
- Usability
  - An updated game board should be printed to the screen after every move.
  - A prompt should display which players turn it is, and it should ask where the player wants to move next.
  - The player should also be asked if he would like to play again.
- Reliability
  - A prompt should also display stating that a column is full if a player tries to put a token in that column.
  - A prompt should also display if the player tries to input an invalid column number.
  - A prompt should display if the player inputs and invalid board size or invalid winning token amount.
- Performance
  - The game should keep track of two players data.
  - The data should be handled in the most efficient way possible.
- Supportability
  - The game should be playable between two players.
  - The program can run on any computer with Java installed.
- Implementation
  - The program must be coded in Java.
  - The program must be able to run on Unix.
  - Comments should be added to better understand the code.
  - Javadoc comments should also be added to know the requirements of the classes and functions.
  - The program should also use an interface for the methods of the class and have an interface specification.

Diagrams

Class Diagrams

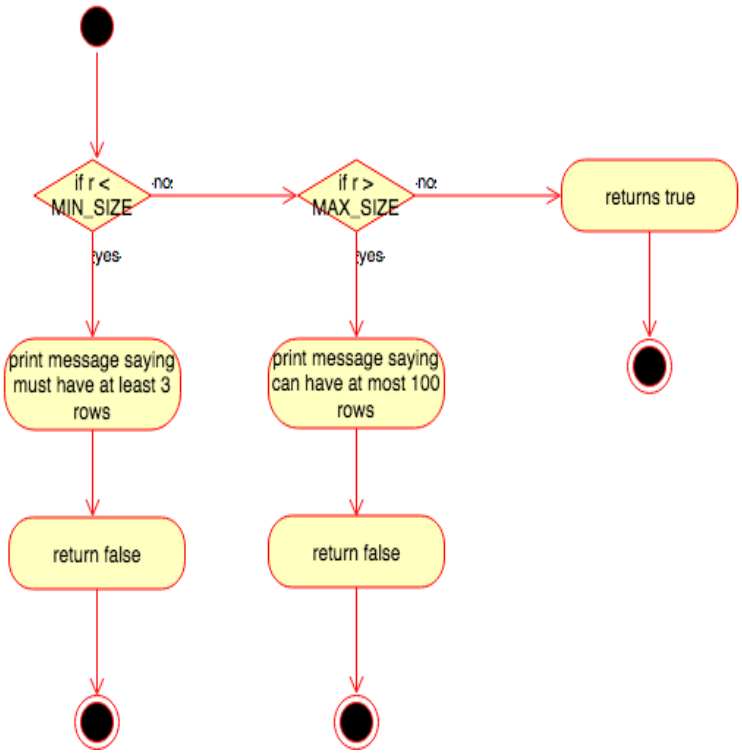


Main

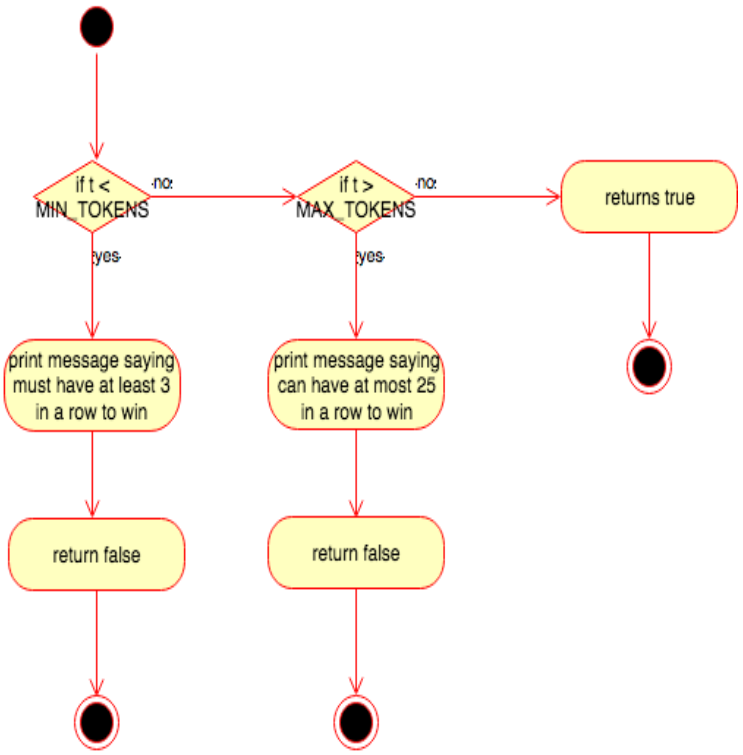

Main
variables :
char playAgain = 'y'
int turns = 0
char play again tracks if the player wishes to play again
turns keep track of how may turns have been made
while playAgain == y
no
yes
turns = 0
ask for how many rows the board should have
rows = user input
while !validateNumRows(rows)
no
yes
ask for how many rows the board should have
rows = user input
ask for how many colums the board should have
columns = user input
while !validateNumColums(columns)
no
yes
ask for how many columns the board should have
columns = user input
ask for how many in a row to win
tokens = user input
while !validateNumTokens(tokens)
no
yes
ask for how many in a row to win
tokens = user input
board = new GameBoard(columns, row, tokens)
while checkForWin == false and checkForTie == false
no
yes
call printTurn
Get move from player X/O
validateColumn
False
True
checkIfFree
False
False
True
Call placeToken
turns++
checkForTie
False
True
print board
Display tie
False
checkForWin
True
print board
Display winner
Play Again?
Yes
No
playAgain = y or playAgain = n
while playAgain != 'y' || playAgain !=
yes
no
playAgain = y or playAgain = n

Private Helper Functions in Main

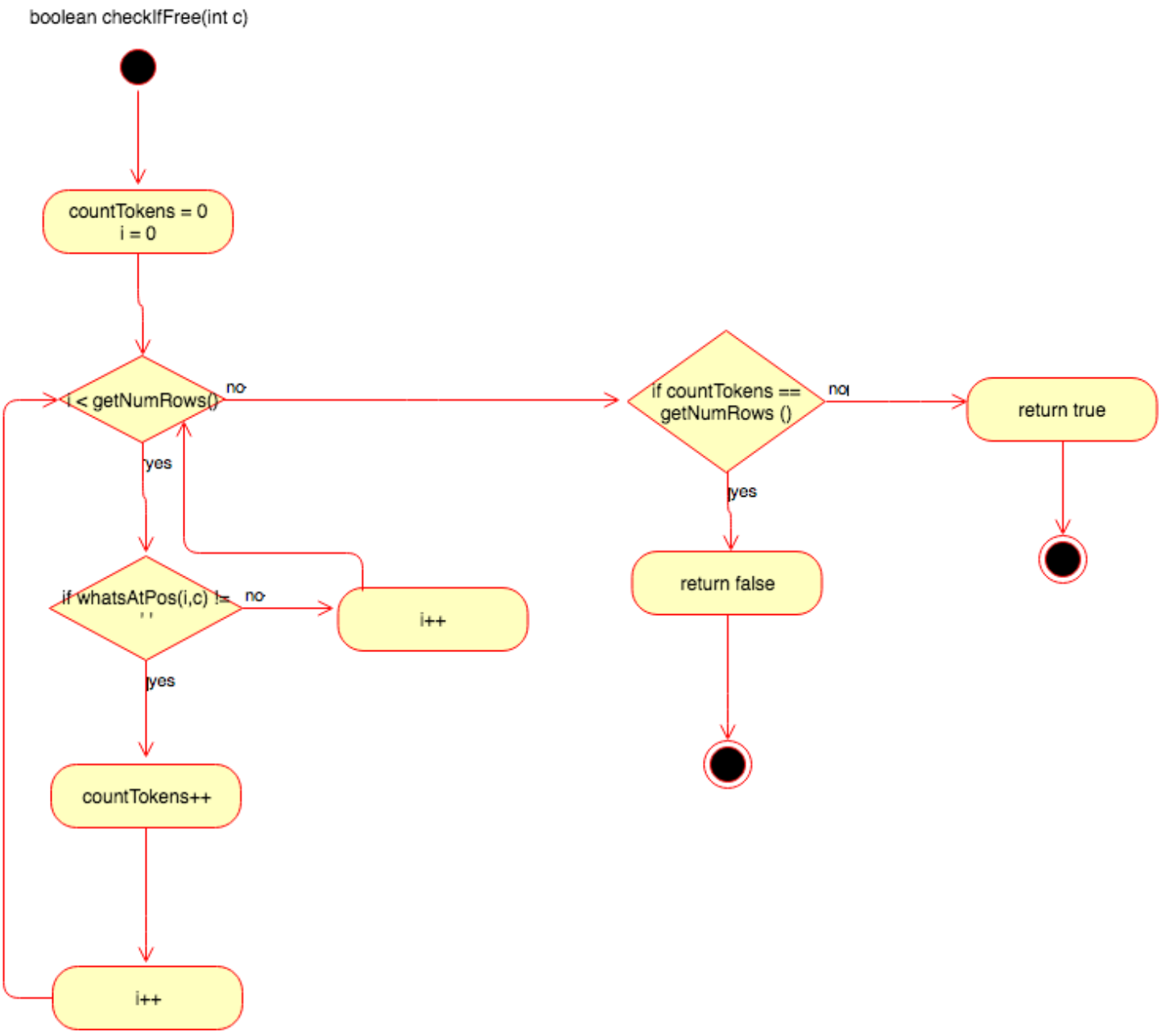
boolean validateNumRows(int r)



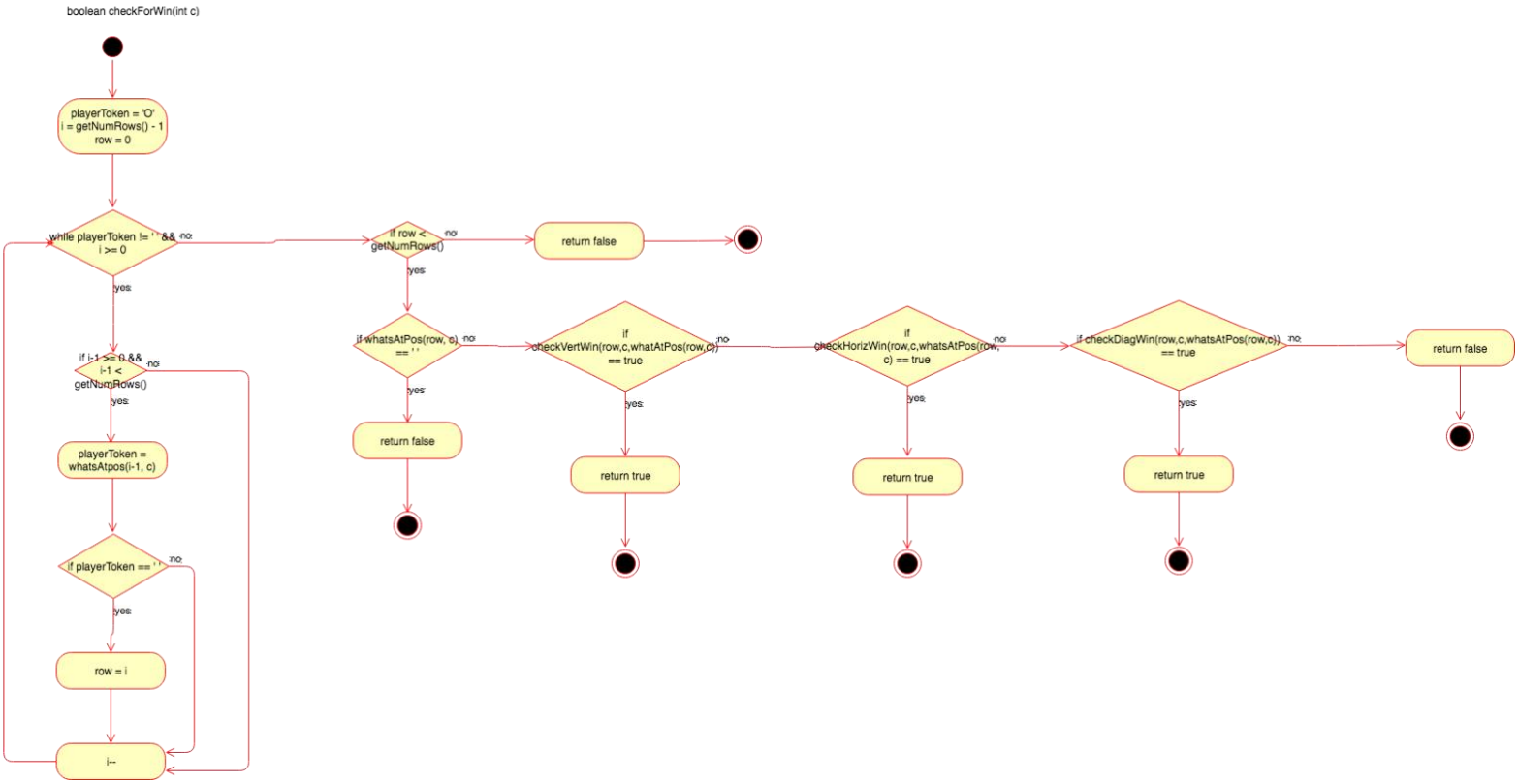
boolean validateNumTokens(int t)



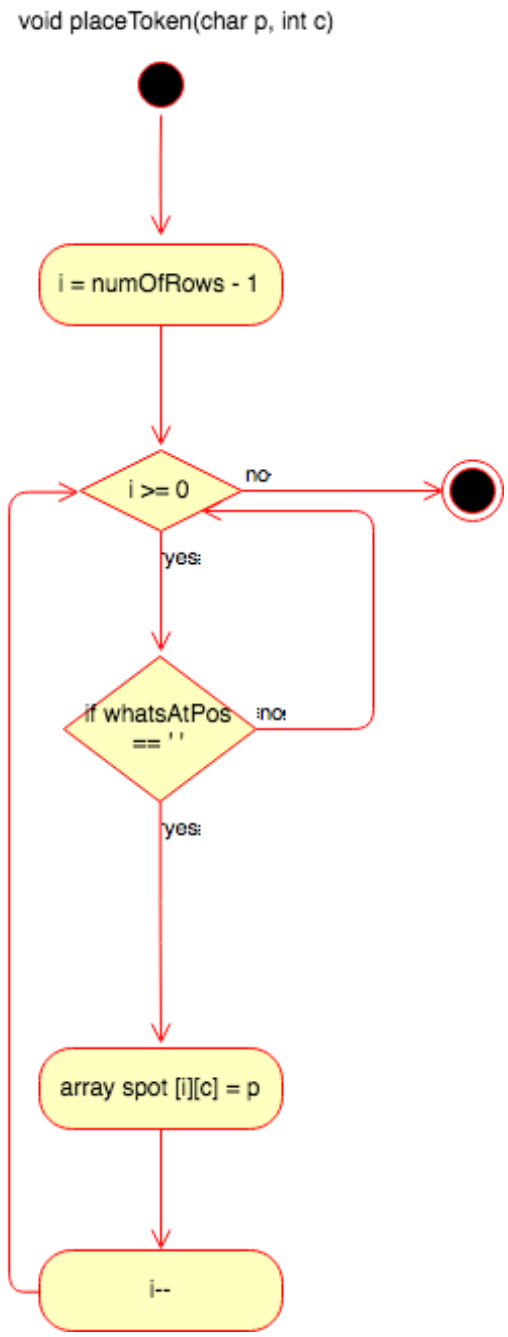
checkIfFree



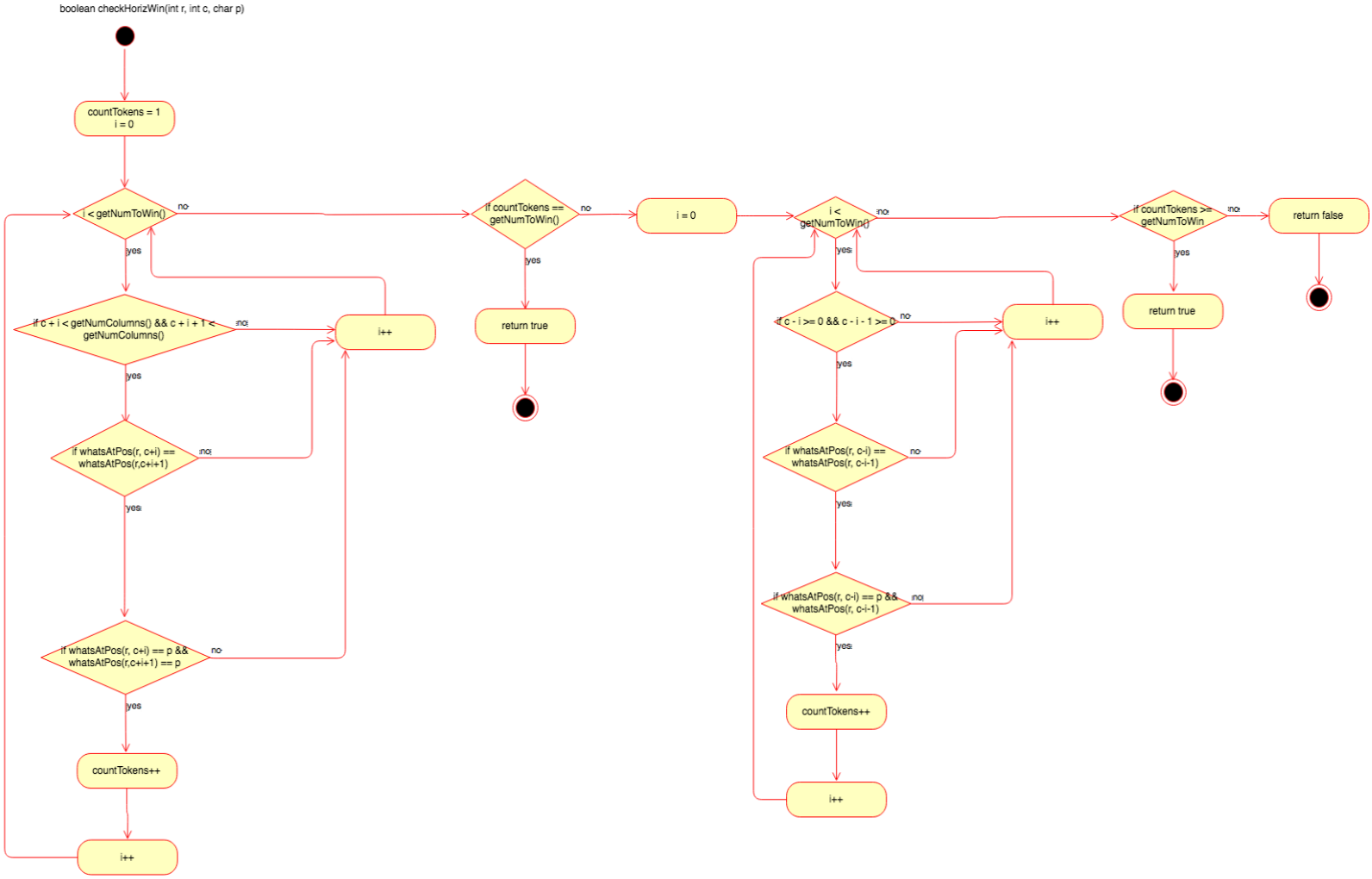
checkForWin



placeToken

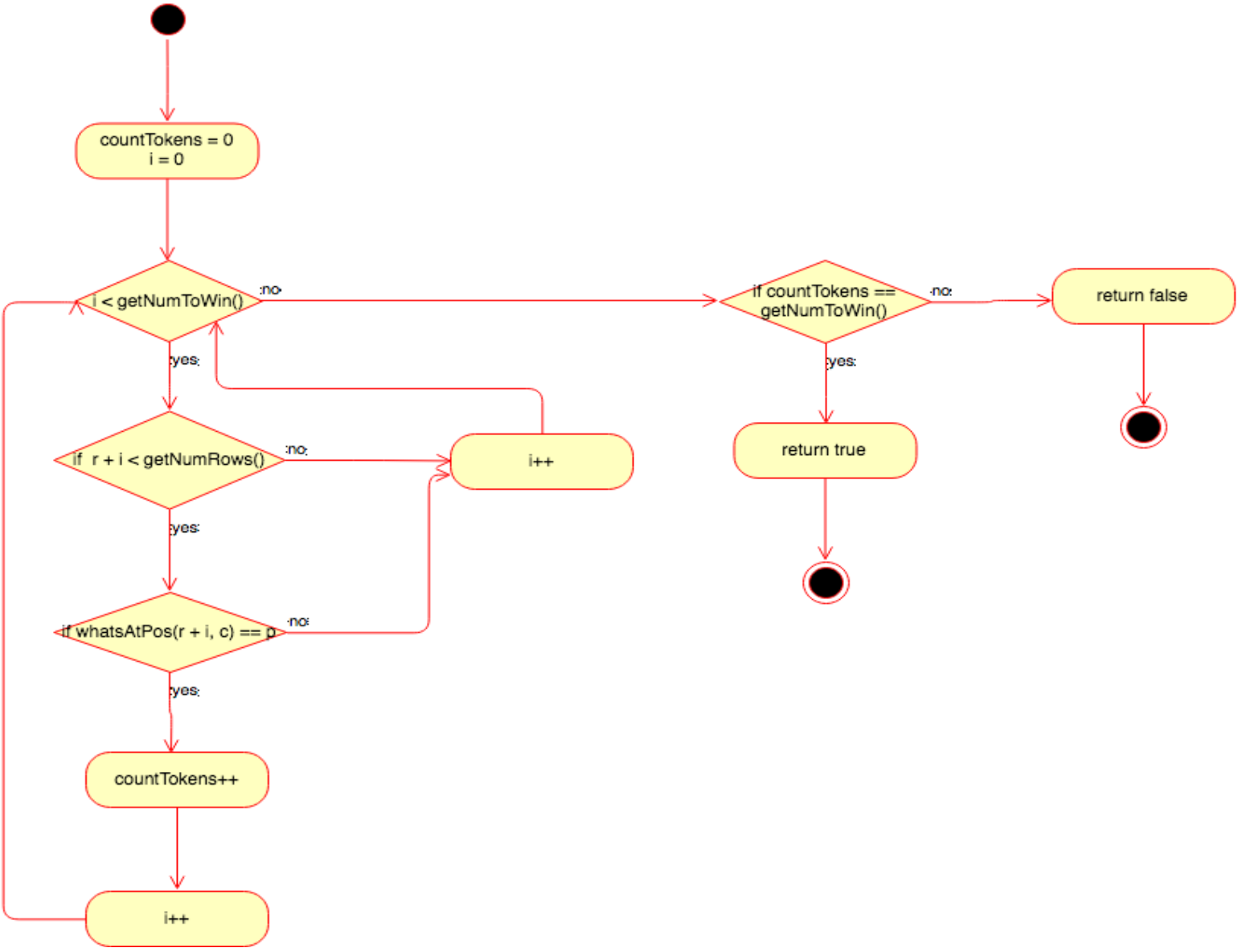

void placeToken(char p, int c)
i = numOfRows - 1
i >= 0
no
yes
if whatsAtPos == ' '
no
yes
array spot [i][c] = p
i--

checkHorizWin

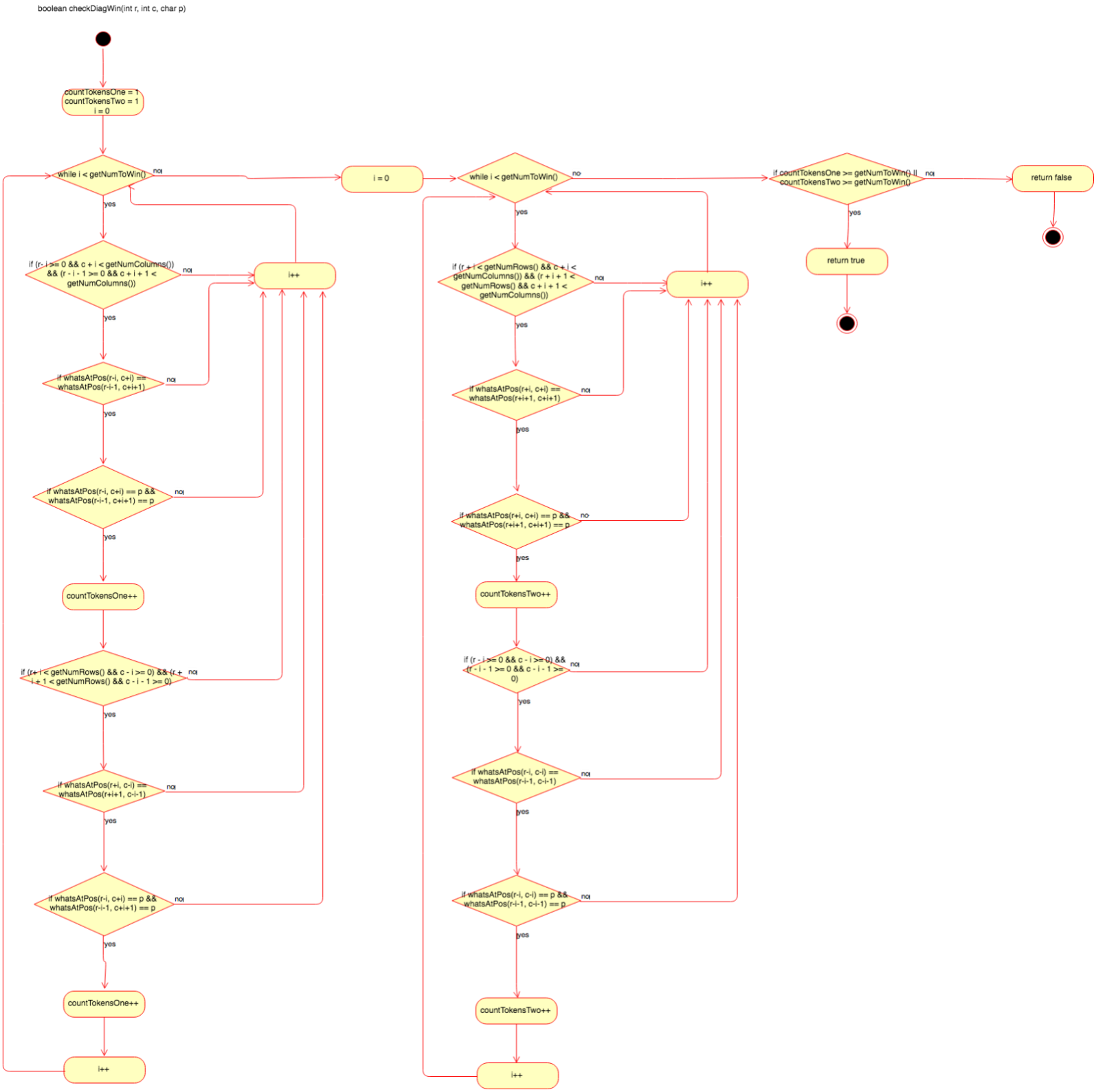


checkVertWin

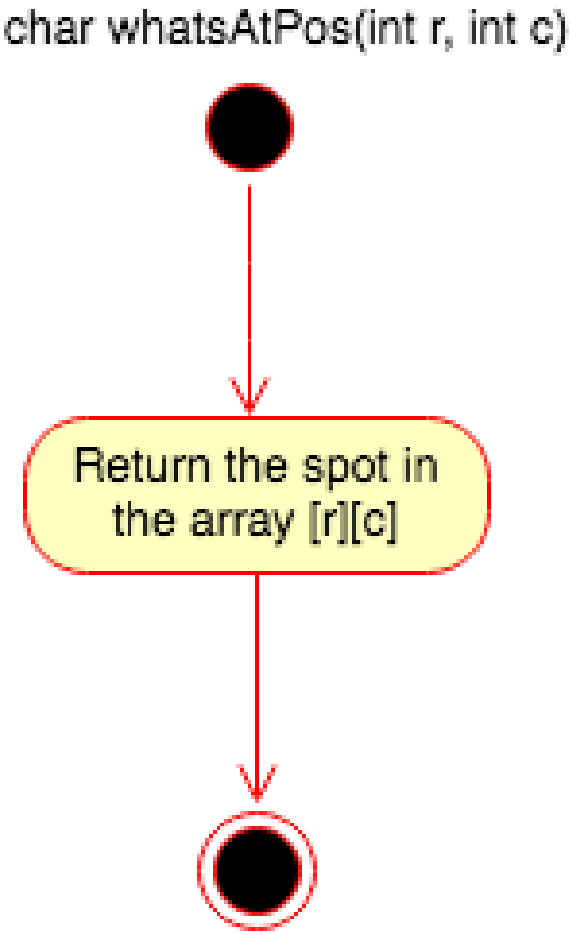
boolean checkVertWin(int r, int c, char p)



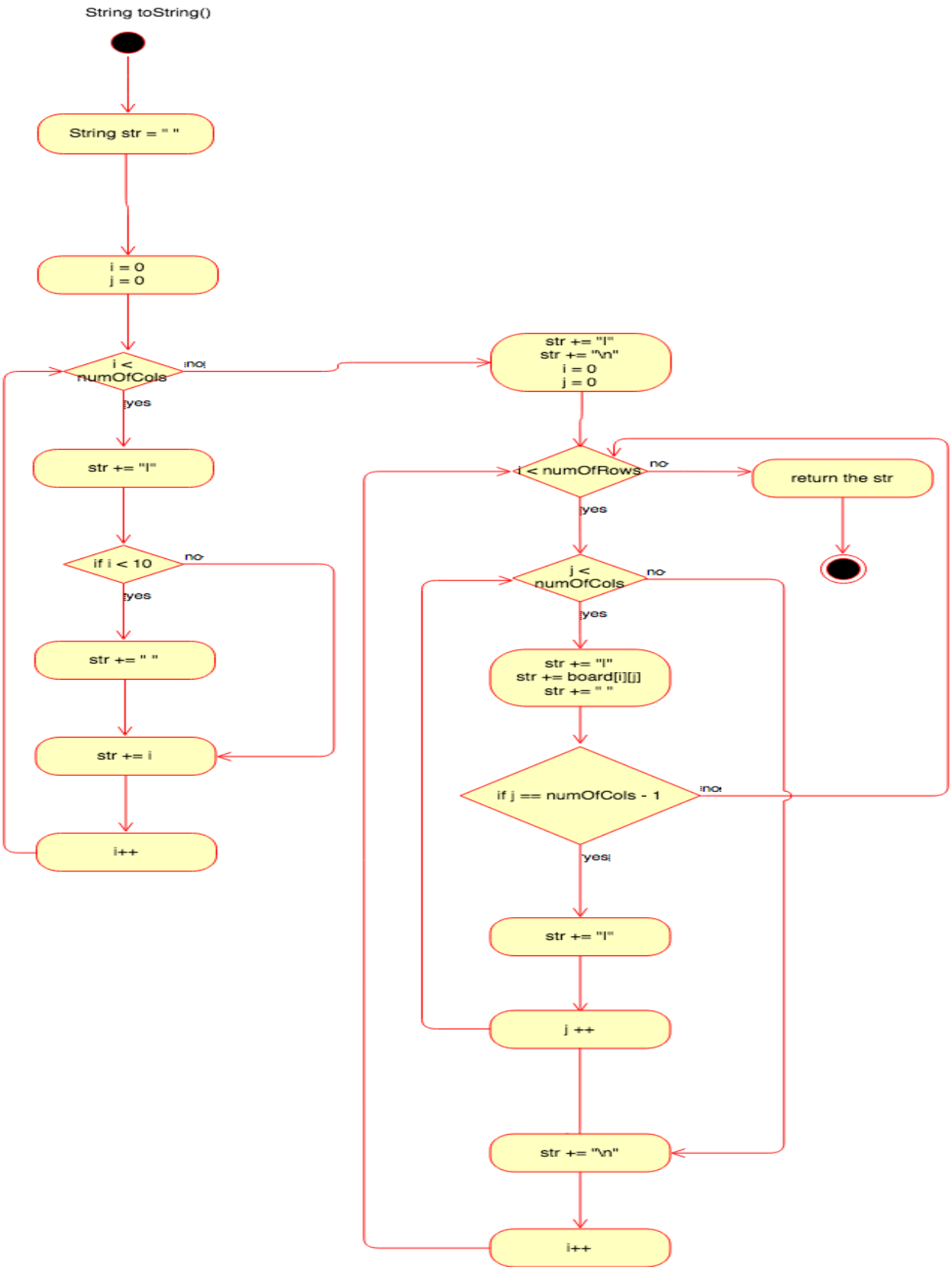
checkDiagWin



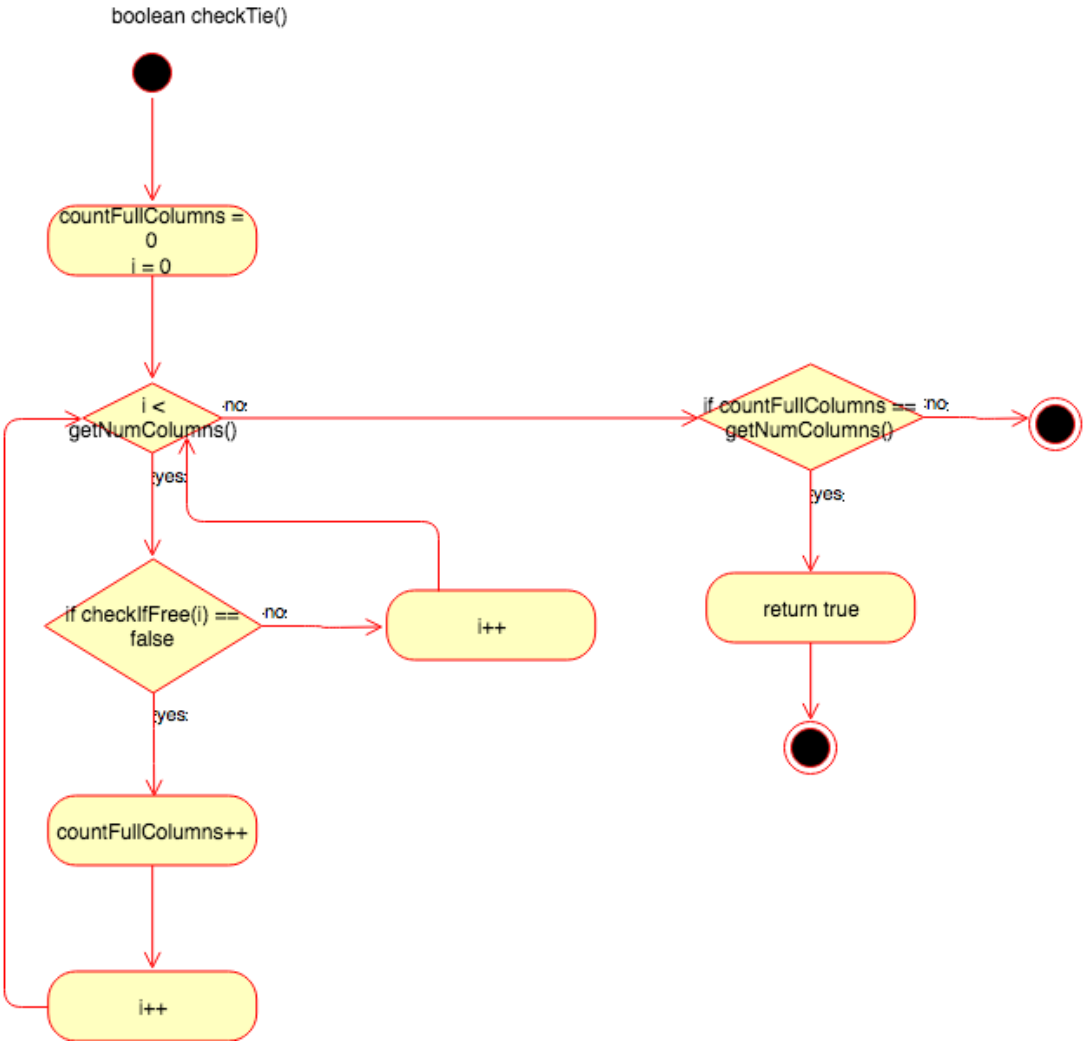
whatsAtPos



toString

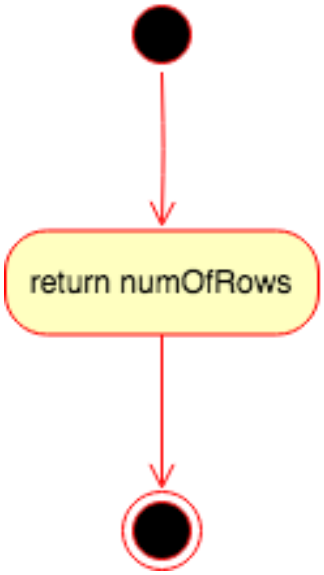

String toString()
String str = " "
i = 0
j = 0
i < numOfCols
no
yes
str += "|"
if i < 10
no
yes
str += " "
str += i
i++
str += "|"
str += "\n"
i = 0
j = 0
i < numOfRows
no
yes
return the str
j < numOfCols
no
yes
str += "|"
str += board[i][j]
str += " "
if j == numOfCols - 1
no
yes
str += "|"
j ++
str += "\n"
i++

checkTie

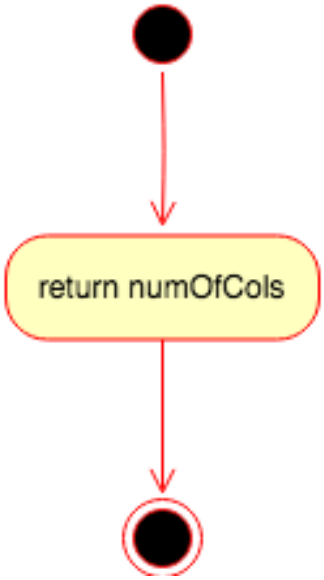

boolean checkTie()
countFullColumns = 0
i = 0
i < getNumColumns()
no:
yes:
if checkIfFree(i) == false
no:
i++
yes:
countFullColumns++
i++
if countFullColumns == getNumColumns()
no:
yes:
return true

Getters

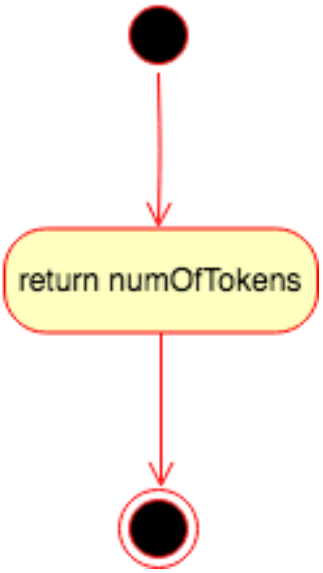
int getNumRows()



int getNumColumns()



int getNumToWin()



Deployment

In order to run my program, you first have to unzip the PaigeClemonsHW2 folder. Then navigate to that folder in your terminal. Once you in the PaigeClemonsHW1 folder, there should only be a cpsc2150 folder, the makefile, and my report in the folder. After you check that stay in that folder, you can compile my program by typing "make" into the terminal. Then to run the program all you have to do is type "make run" then you can play the Connect Four game as usual. If you wish to delete the .class files then all you have to do is type "make clean" into the terminal.