JEREMY HOLLOWAY

10/28/2018

Technical Report

Prepared for Kevin Plis

Prepared by Jeremy Holloway

CPSC-2150-001

Fall 2018

TABLE OF CONTENTS

1. REQUIREMENT ANALYSIS

User Story:

- As a user, I can choose which column to place a token to win the game.
- As a user, I will be notified when it is my turn so I can place a marker.
- As a user, I will be notified of a win so I can rub it in my opponents face.
- As a user, I will be notified if I make an incorrect move so I may take another turn.
- As a user, I will be able to choose how many players can play the game.
- As a user, I will be able to select what character I want to play with.
- As a user, I will be notified if my selected token is already in use.
- As a user, I will be able to choose if I want to play a memory efficient game or a fast game.

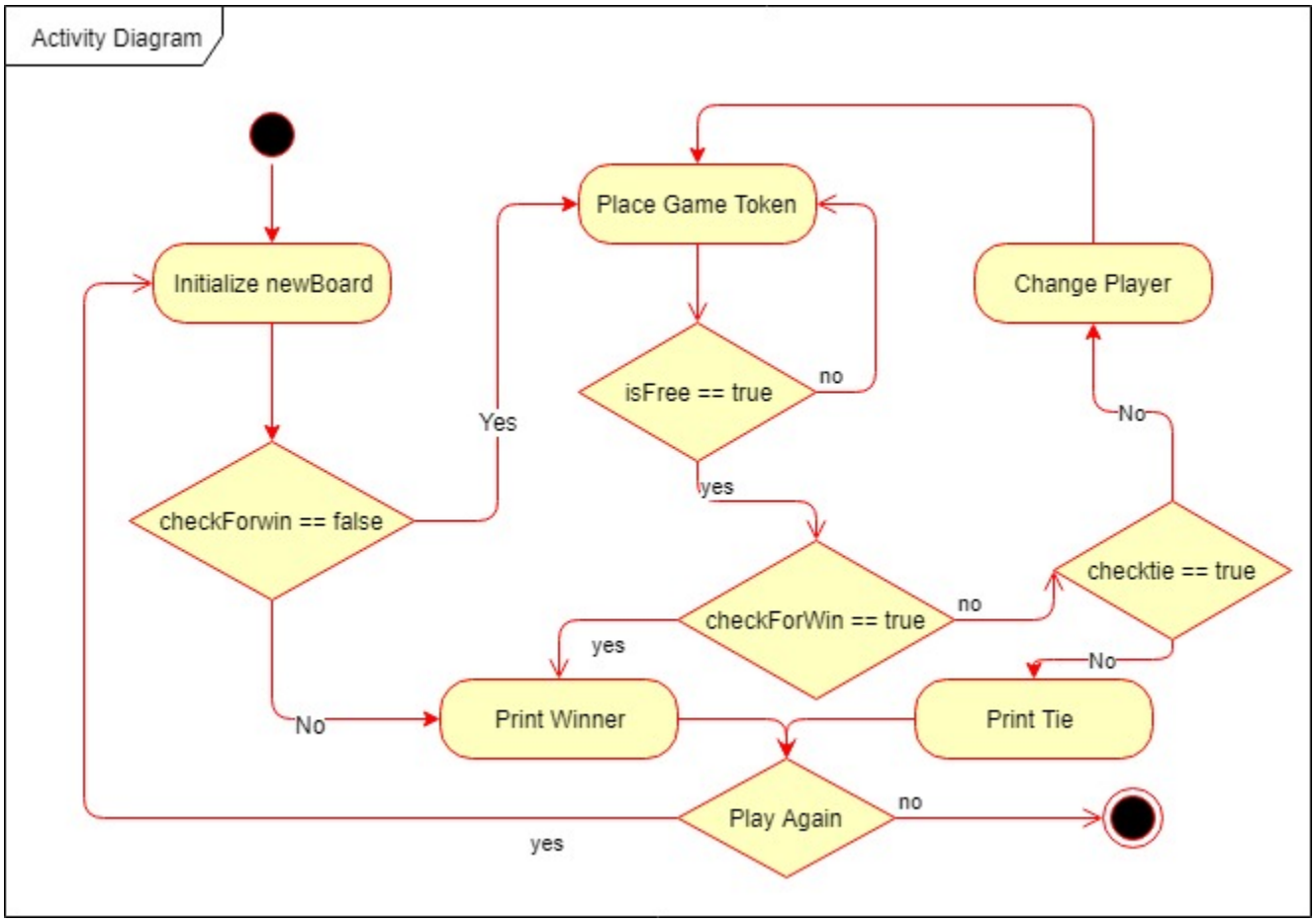
Functional Requirements:

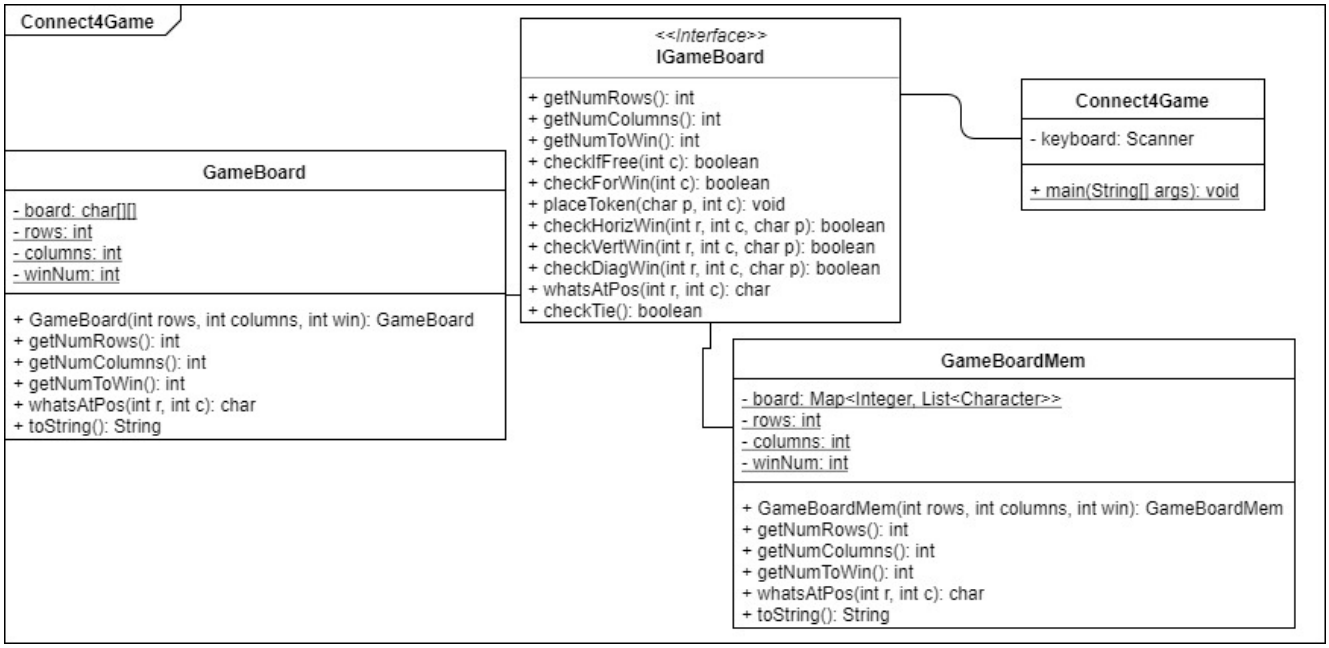
- A user will be able to choose which column to place their token.
- A user will be able to choose to play again after winning, losing, or drawing.
- A user will be able to choose the number of players.
- A user will be able to choose which version of the game to play.

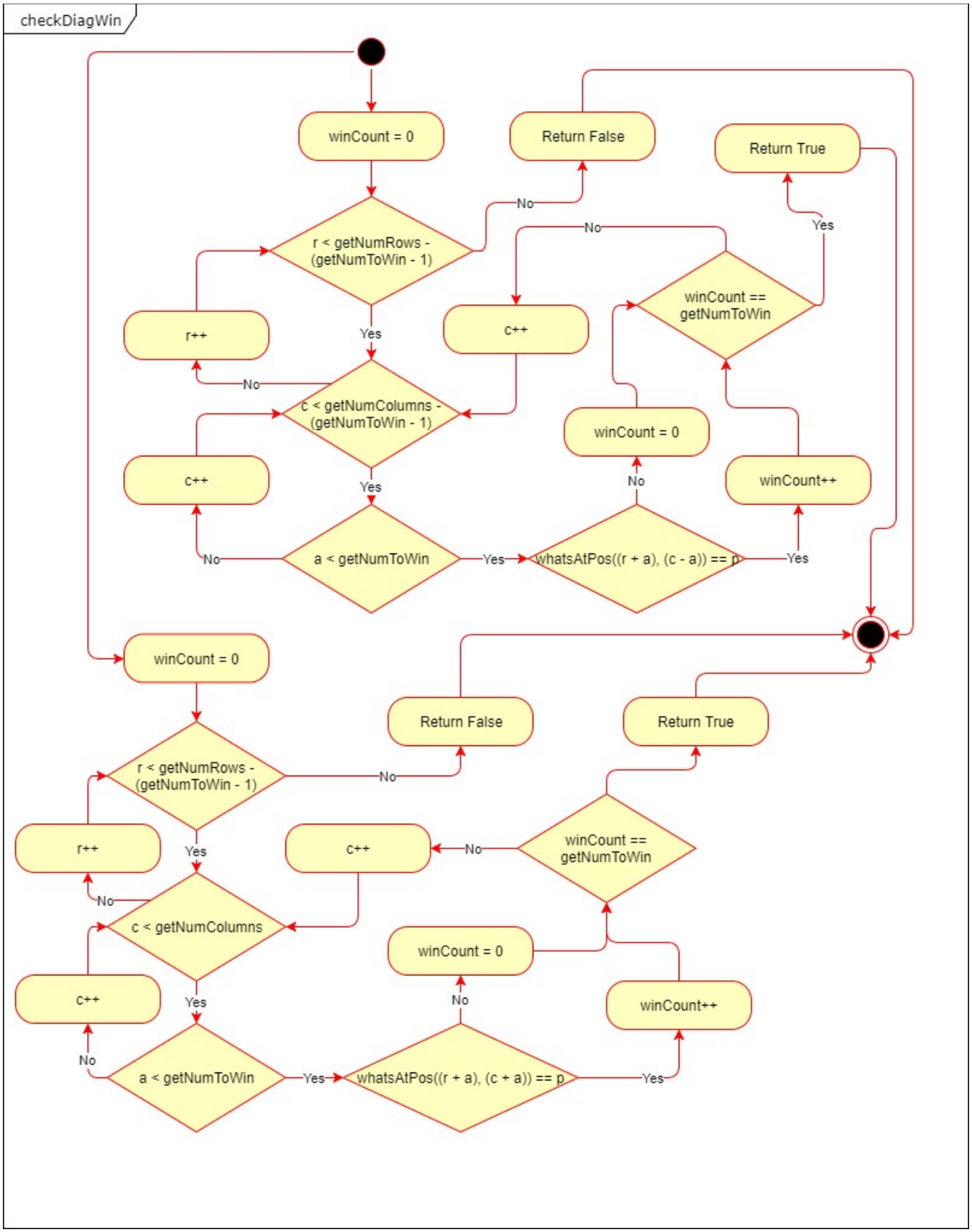
Non-Functional Requirements:

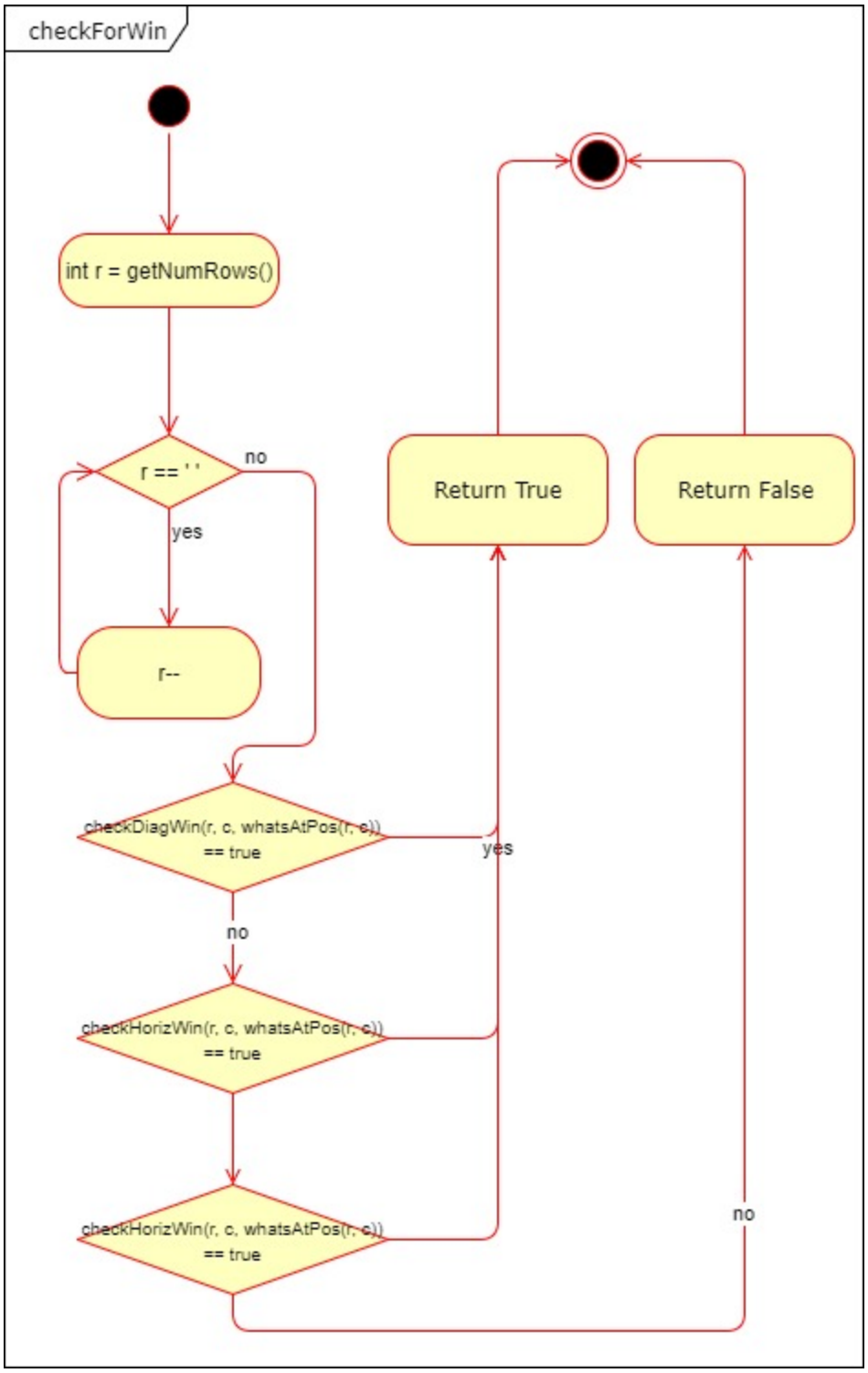
- The program will automatically determine if a winning move is made after each turn or if a draw has occurred.
- The program will be implemented using a 2D character array or a hash map to represent the gameboard.
- The program will have an instant response time.
- The program is modular. If the client desires to remove certain functions, they may be commented out. These methods will work on any system with JDK8 installed.
- The program is written in Java. The system must have JDK8 installed.
- The program was designed for a Unix based environment.
- This program will use a makefile with the commands; make, make run, and make clean.
- All interfaces between the user and the app will be displayed via the terminal window.

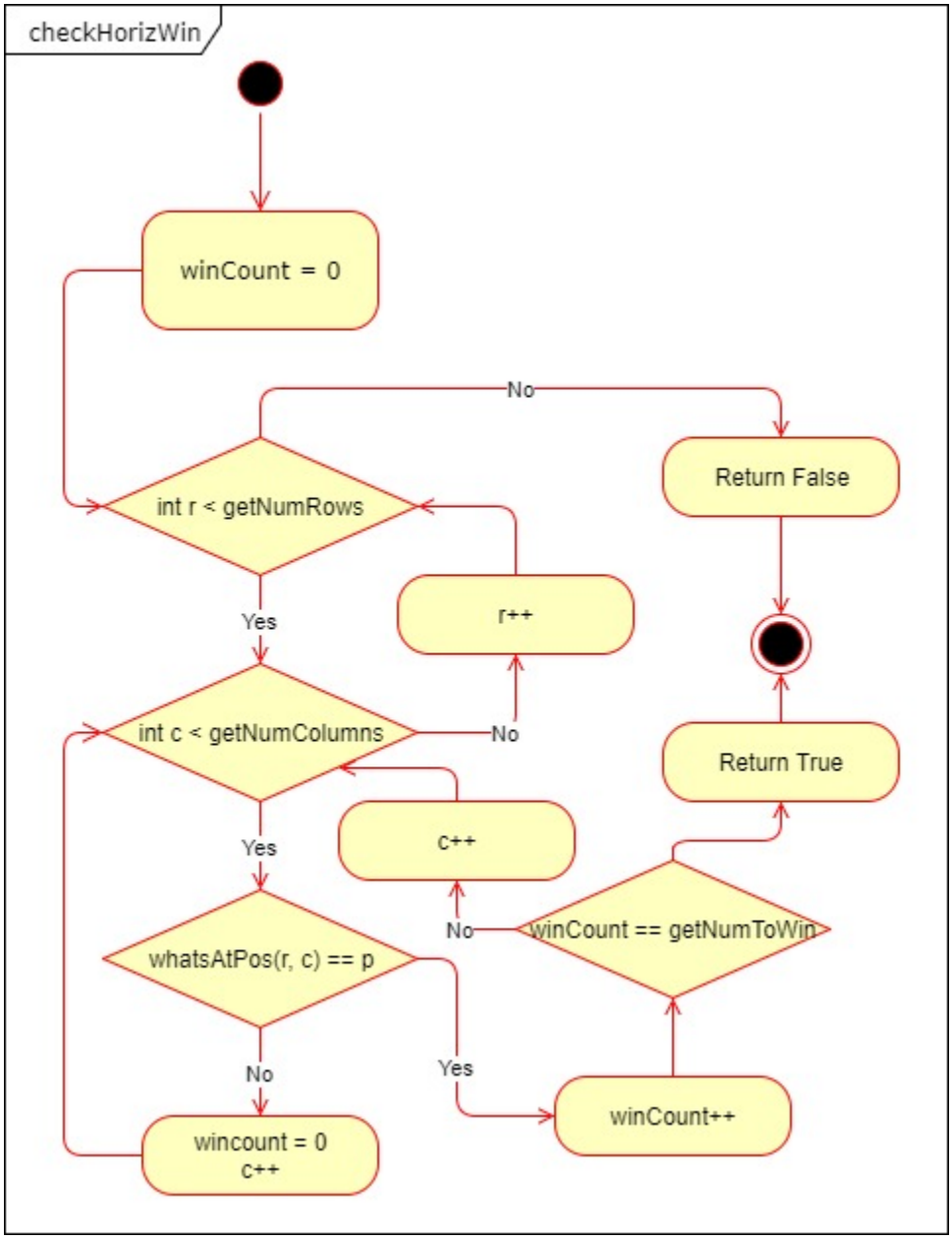
2. DESIGN

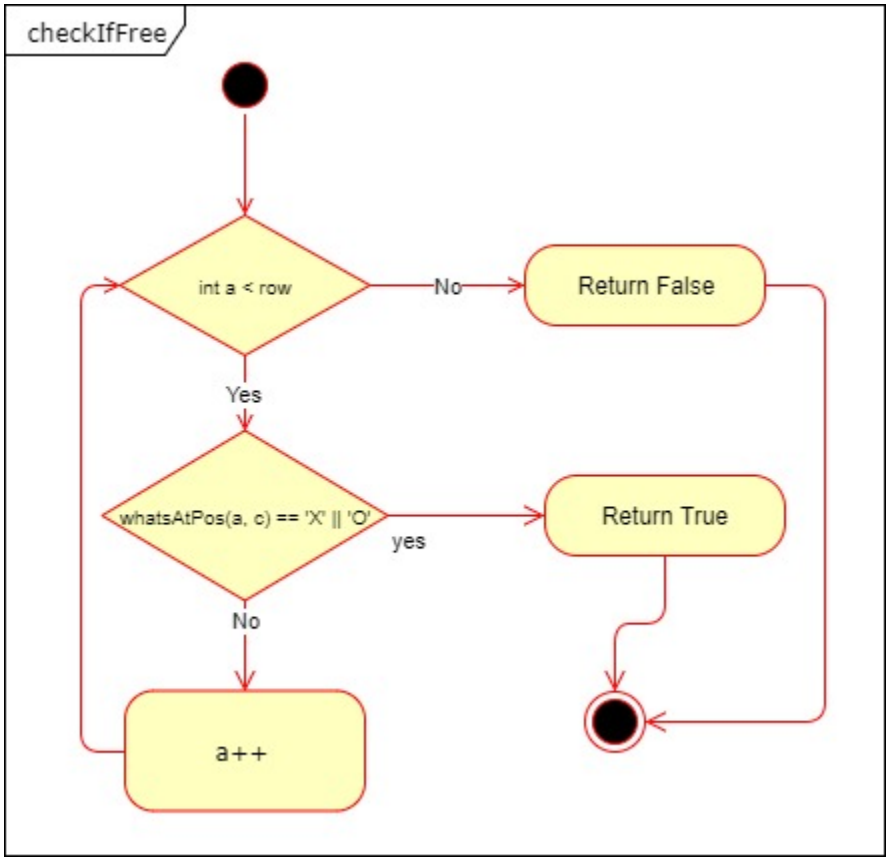

Activity Diagram
Initialize newBoard
Place Game Token
Change Player
isFree == true
no
yes
Yes
No
checkForwin == false
checktie == true
checkForWin == true
no
yes
No
No
Print Winner
Print Tie
Play Again
no
yes

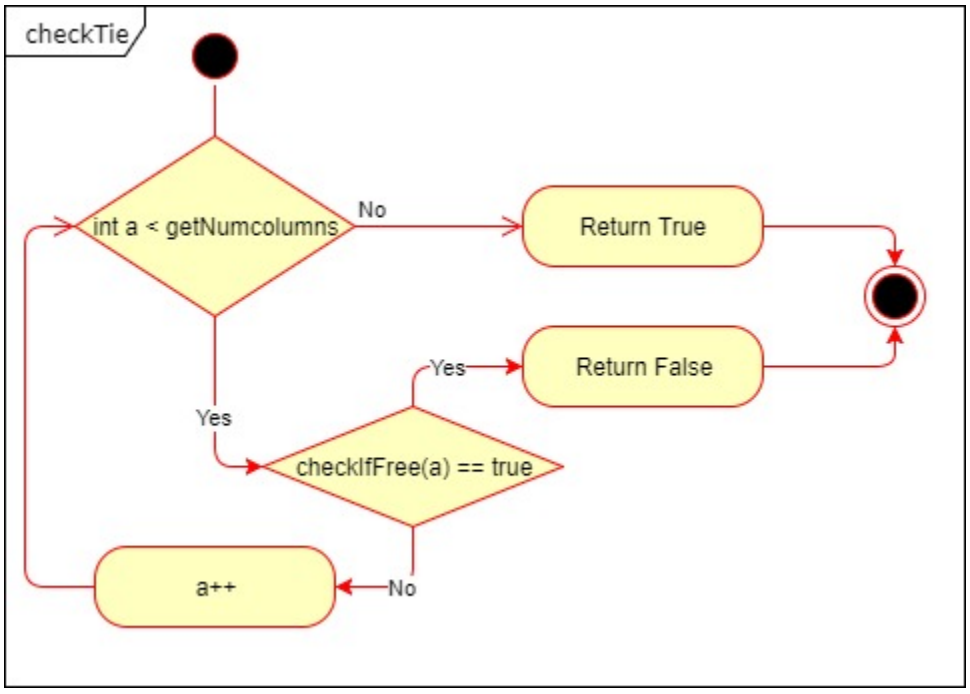

Connect4Game
<<Interface>>
IGameBoard
+ getNumRows(): int
+ getNumColumns(): int
+ getNumToWin(): int
+ checkIfFree(int c): boolean
+ checkForWin(int c): boolean
+ placeToken(char p, int c): void
+ checkHorizWin(int r, int c, char p): boolean
+ checkVertWin(int r, int c, char p): boolean
+ checkDiagWin(int r, int c, char p): boolean
+ whatsAtPos(int r, int c): char
+ checkTie(): boolean
Connect4Game
- keyboard: Scanner
+ main(String[] args): void
GameBoard
- board: char[][]
- rows: int
- columns: int
- winNum: int
+ GameBoard(int rows, int columns, int win): GameBoard
+ getNumRows(): int
+ getNumColumns(): int
+ getNumToWin(): int
+ whatsAtPos(int r, int c): char
+ toString(): String
GameBoardMem
- board: Map<Integer, List<Character>>
- rows: int
- columns: int
- winNum: int
+ GameBoardMem(int rows, int columns, int win): GameBoardMem
+ getNumRows(): int
+ getNumColumns(): int
+ getNumToWin(): int
+ whatsAtPos(int r, int c): char
+ toString(): String

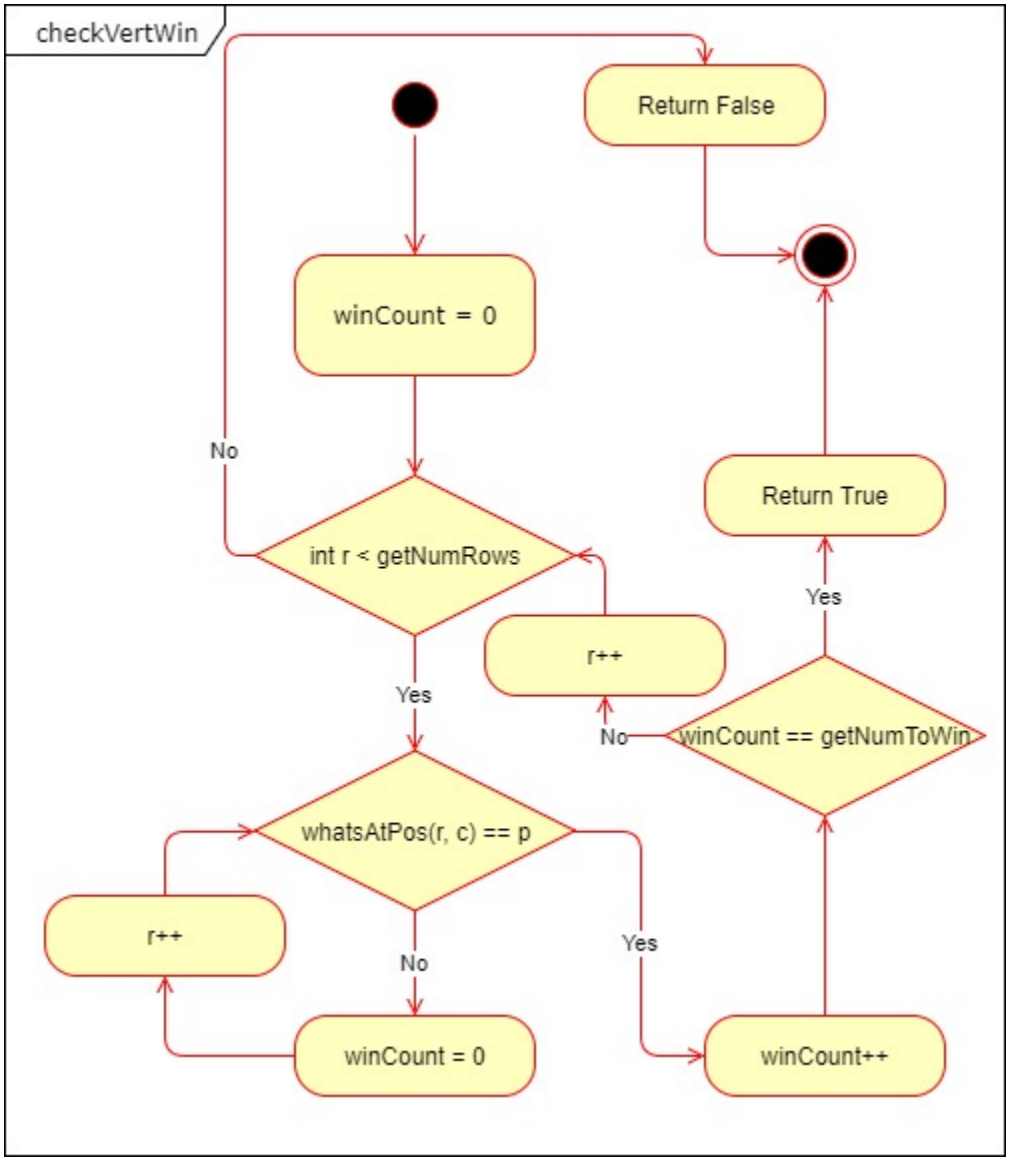

checkDiagWin
winCount = 0
r < getNumRows - (getNumToWin - 1)
No
Return False
Return True
Yes
No
winCount == getNumToWin
Yes
r++
Yes
c++
No
c < getNumColumns - (getNumToWin - 1)
winCount = 0
c++
Yes
No
winCount++
No
a < getNumToWin
Yes
whatsAtPos((r + a), (c - a)) == p
Yes
winCount = 0
Return False
Return True
r < getNumRows - (getNumToWin - 1)
No
r++
Yes
c++
No
winCount == getNumToWin
No
c < getNumColumns
winCount = 0
c++
Yes
No
winCount++
No
a < getNumToWin
Yes
whatsAtPos((r + a), (c + a)) == p
Yes

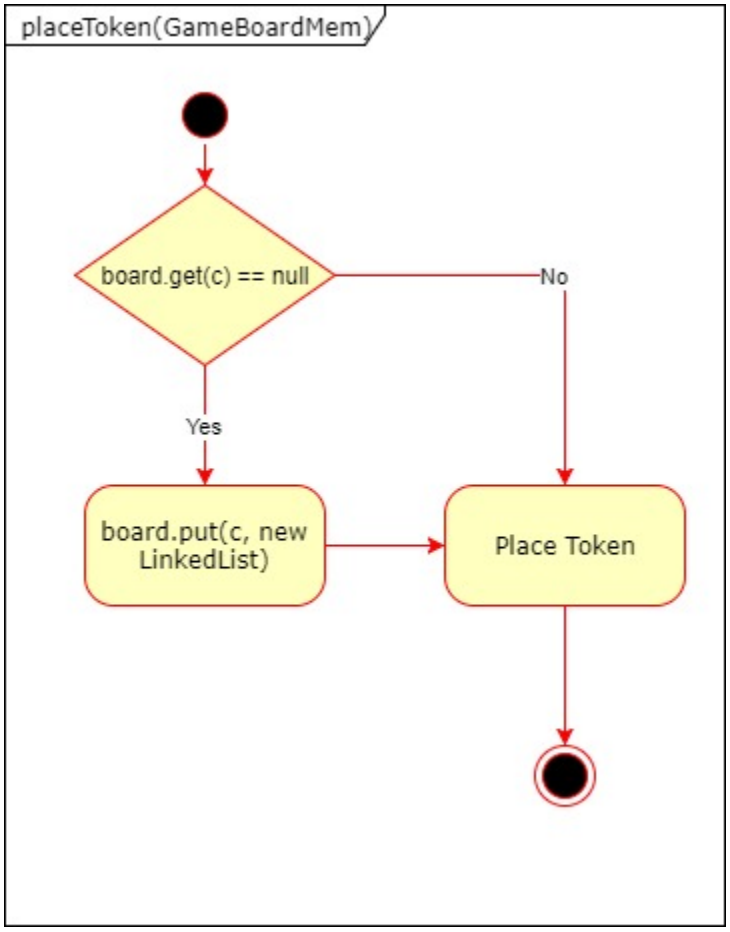

checkForWin
int r = getNumRows()
r == ' '
no
yes
r--
Return True
Return False
checkDiagWin(r, c, whatsAtPos(r, c)) == true
yes
no
checkHorizWin(r, c, whatsAtPos(r, c)) == true
checkHorizWin(r, c, whatsAtPos(r, c)) == true
no

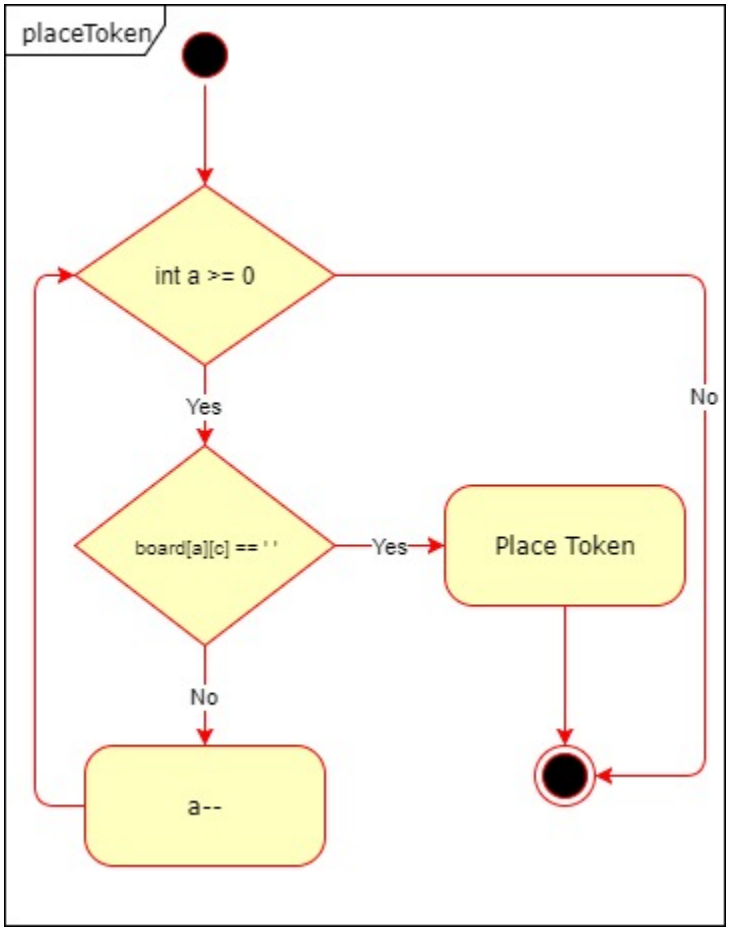

checkHorizWin
winCount = 0
No
Return False
int r < getNumRows
r++
Yes
int c < getNumColumns
No
Return True
c++
Yes
No
winCount == getNumToWin
whatsAtPos(r, c) == p
No
Yes
winCount++
wincount = 0
c++

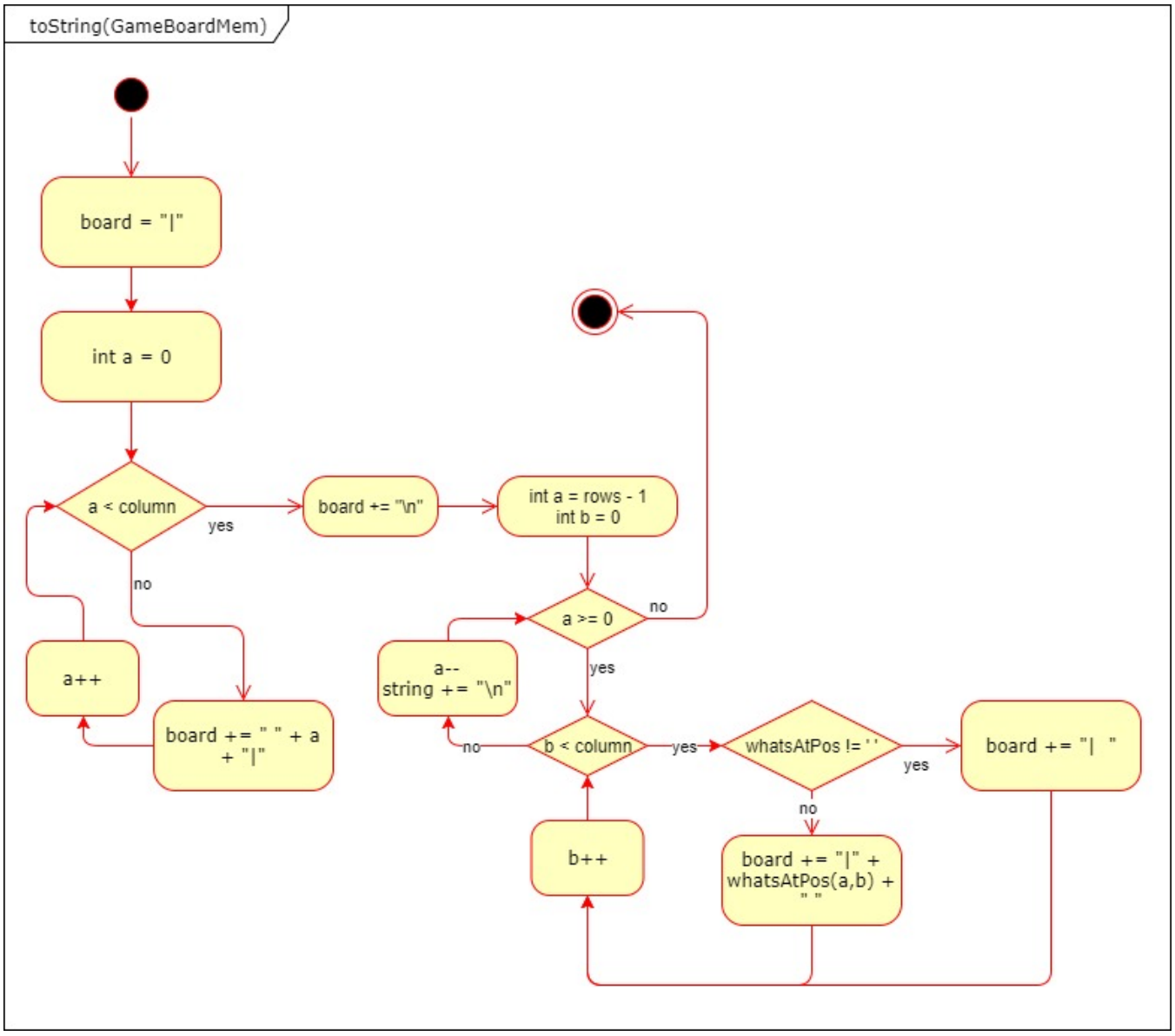

checkIfFree
int a < row
No
Return False
Yes
whatsAtPos(a, c) == 'X' || 'O'
yes
Return True
No
a++

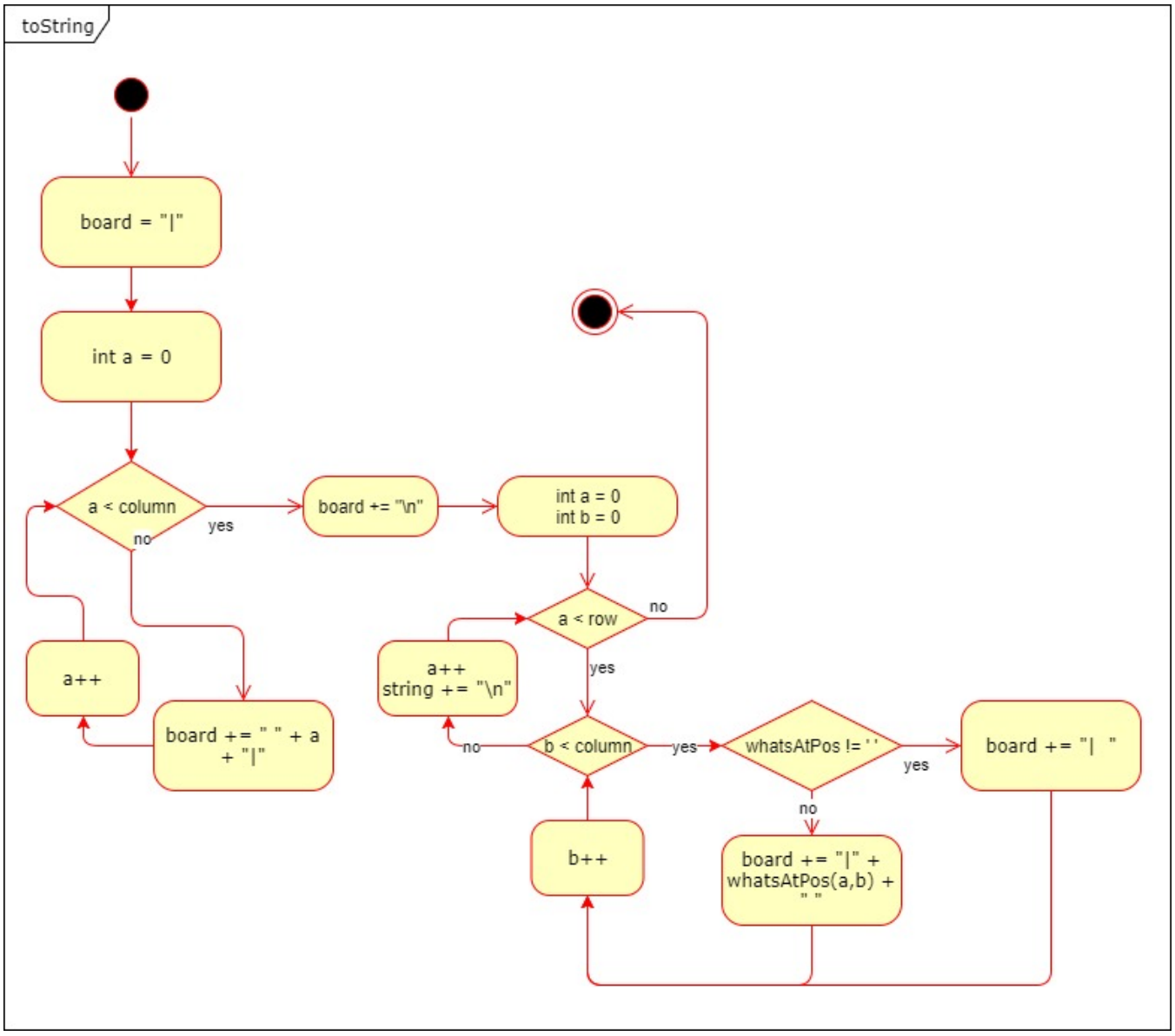

checkTie
int a < getNumcolumns
No
Return True
Yes
Return False
Yes
checkIfFree(a) == true
a++
No


checkVertWin
Return False
winCount = 0
No
Return True
int r < getNumRows
Yes
r++
Yes
No
winCount == getNumToWin
whatsAtPos(r, c) == p
r++
Yes
No
winCount = 0
winCount++


placeToken(GameBoardMem)
board.get(c) == null
No
Yes
board.put(c, new LinkedList)
Place Token


placeToken
int a >= 0
Yes
No
board[a][c] == ' '
Yes
Place Token
No
a--


toString(GameBoardMem)
board = "|"
int a = 0
a < column
yes
no
board += "\n"
int a = rows - 1
int b = 0
a >= 0
no
yes
a++
board += " " + a + "|"
a--
string += "\n"
b < column
yes
no
whatsAtPos != ' '
yes
no
board += "| "
b++
board += "|" + whatsAtPos(a,b) + " "


toString
board = "|"
int a = 0
a < column
yes
no
board += "\n"
int a = 0
int b = 0
a < row
no
yes
a++
board += " " + a + "|"
a++
string += "\n"
b < column
yes
no
whatsAtPos != ' '
yes
board += "| "
no
b++
board += "|" + whatsAtPos(a,b) + " "

4. DEPLOYMENT

This program will include a make file which can be used for three operations make, make run, make clean. The makefile must be in the same folder as cpsc2150 and not inside of it or any other subfolder.

make:

Expected output:

javac cpsc2150/connectX/Connect4Game.java

make run:

Expected output:

java cpsc2150.connectX.Connect4Game

start of game

make clean:

Expected output:

rm -f *.class