Project 1: ConnectX – Part 1

Author: Kevin Mody

Class: CPSC 2150/2151

User Story:

Functional Requirements:

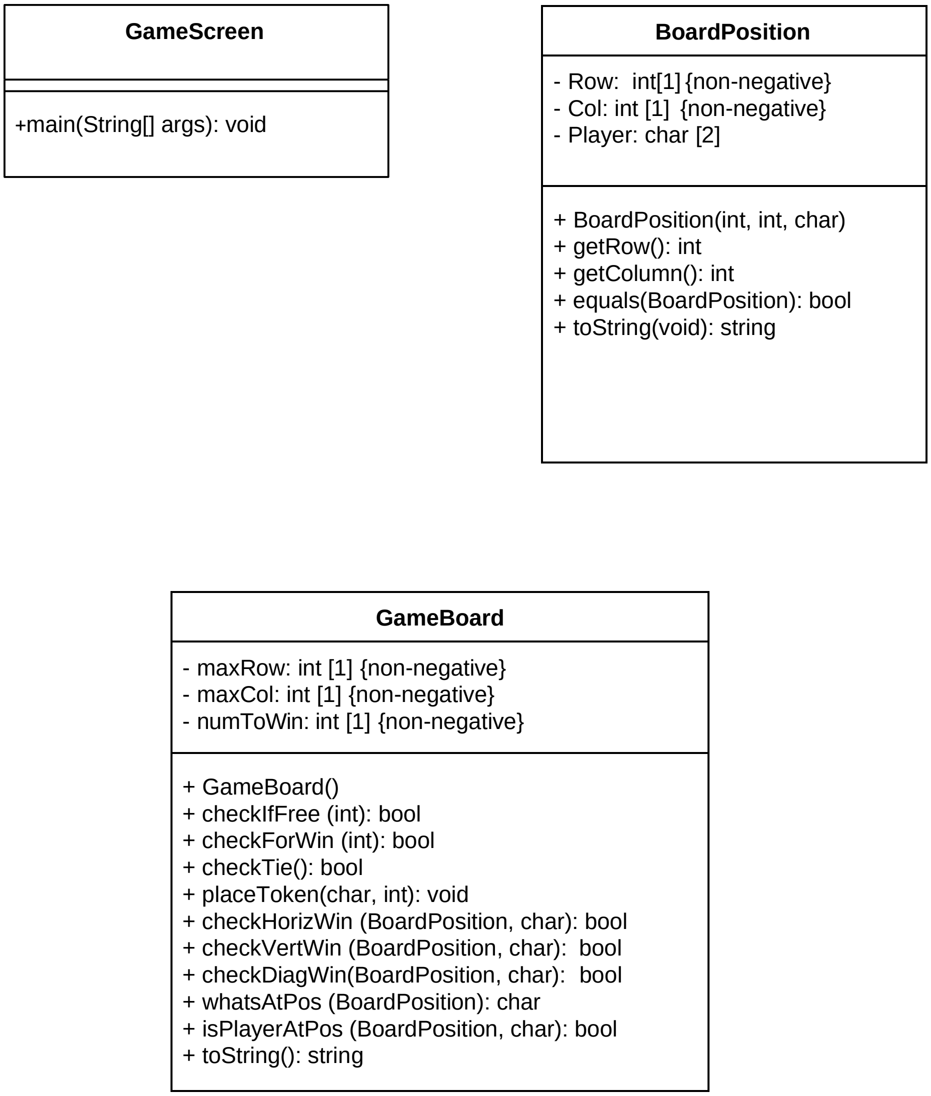
1. As a user, I go first as ‘X’ and second player as ‘O’.
2. As a user, I can start from any given columns.
3. As a user, I need 5 in row horizontally, vertically, or diagonally to win.
4. As a user, I can place ‘X’ token in any columns.
5. As a user, I can place ‘O” token in any columns.
6. As a user, I can try to stop other player winning by placing the token in the columns.
7. As a user, I make a move right after second player ‘O’ or ‘X’ makes the move.
8. As a user If I win, I get decide to play again or exit the game.
9. As a user if the game ties, I have an option to play it again or close it.
10. As a user, to win I only have to get ‘X’s or ‘O’s in row, once only.
11. As a user, if I put a token in a column(s) that is/are already full, it will inform me an error and ask me to choose again.

Non-Functional Requirements:

1. The program must be coded in Java.
2. The program must be able to run on Unix, Windows and MacOS computers.
3. The program must be able to run SoC servers/computers.
4. Time for printing game board bust be quick.
5. Time for inputting tokens must be efficient and fast.
6. Time to load a new game must be quick.

UML Class Diagram

ConnectX - UML Classes



GameScreen

+main(String[] args): void

BoardPosition

- Row: int[1] {non-negative}
- Col: int [1] {non-negative}
- Player: char [2]

+ BoardPosition(int, int, char)
+ getRow(): int
+ getColumn(): int
+ equals(BoardPosition): bool
+ toString(void): string

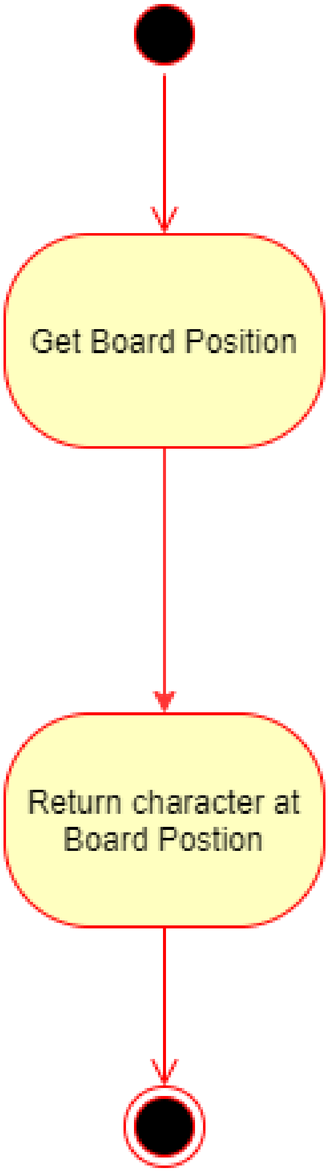
GameBoard

- maxRow: int [1] {non-negative}
- maxCol: int [1] {non-negative}
- numToWin: int [1] {non-negative}

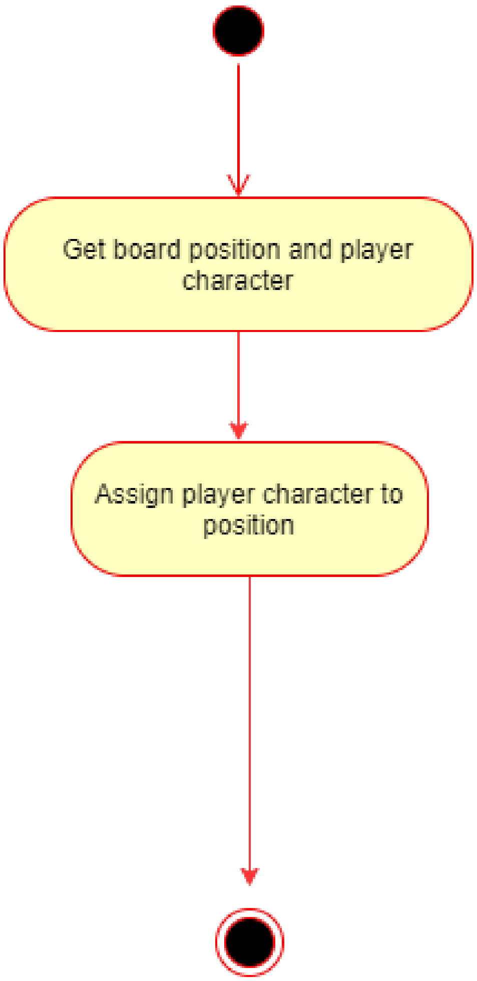
+ GameBoard()
+ checkIfFree (int): bool
+ checkForWin (int): bool
+ checkTie(): bool
+ placeToken(char, int): void
+ checkHorizWin (BoardPosition, char): bool
+ checkVertWin (BoardPosition, char): bool
+ checkDiagWin(BoardPosition, char): bool
+ whatsAtPos (BoardPosition): char
+ isPlayerAtPos (BoardPosition, char): bool
+ toString(): string

UML Activity Diagrams

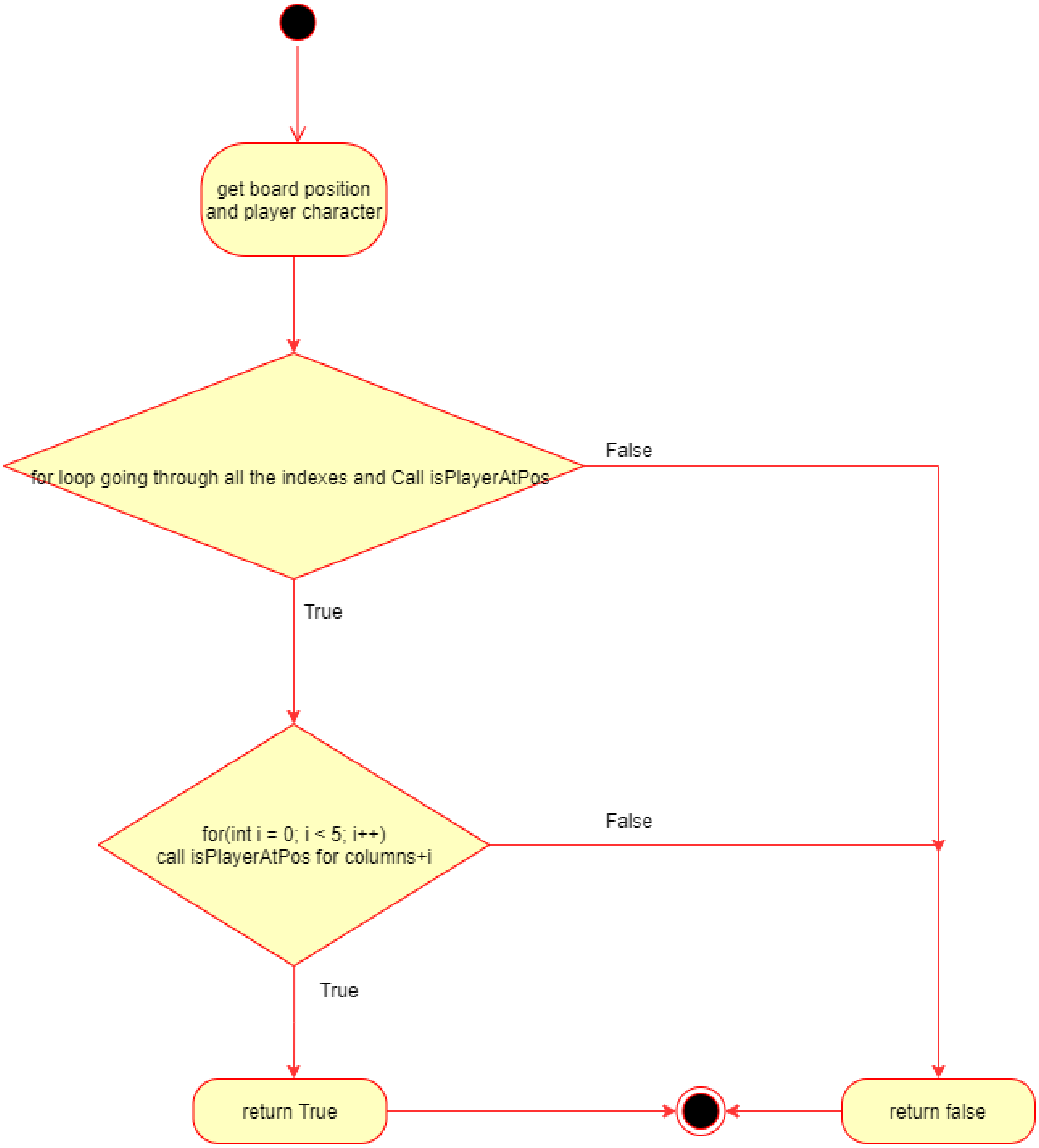
whatAtPos



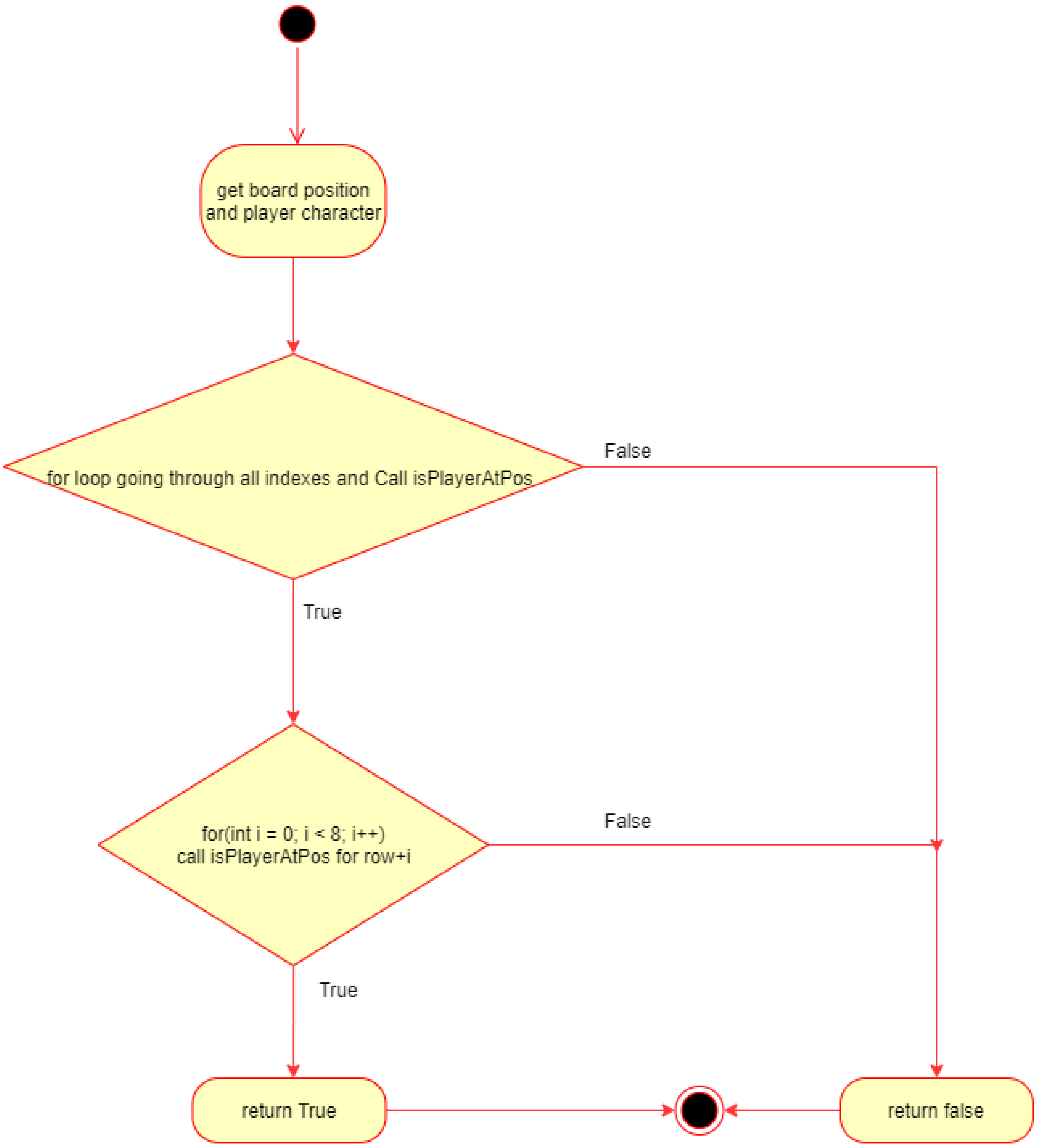
placeToken


Get board position and player character
Assign player character to position

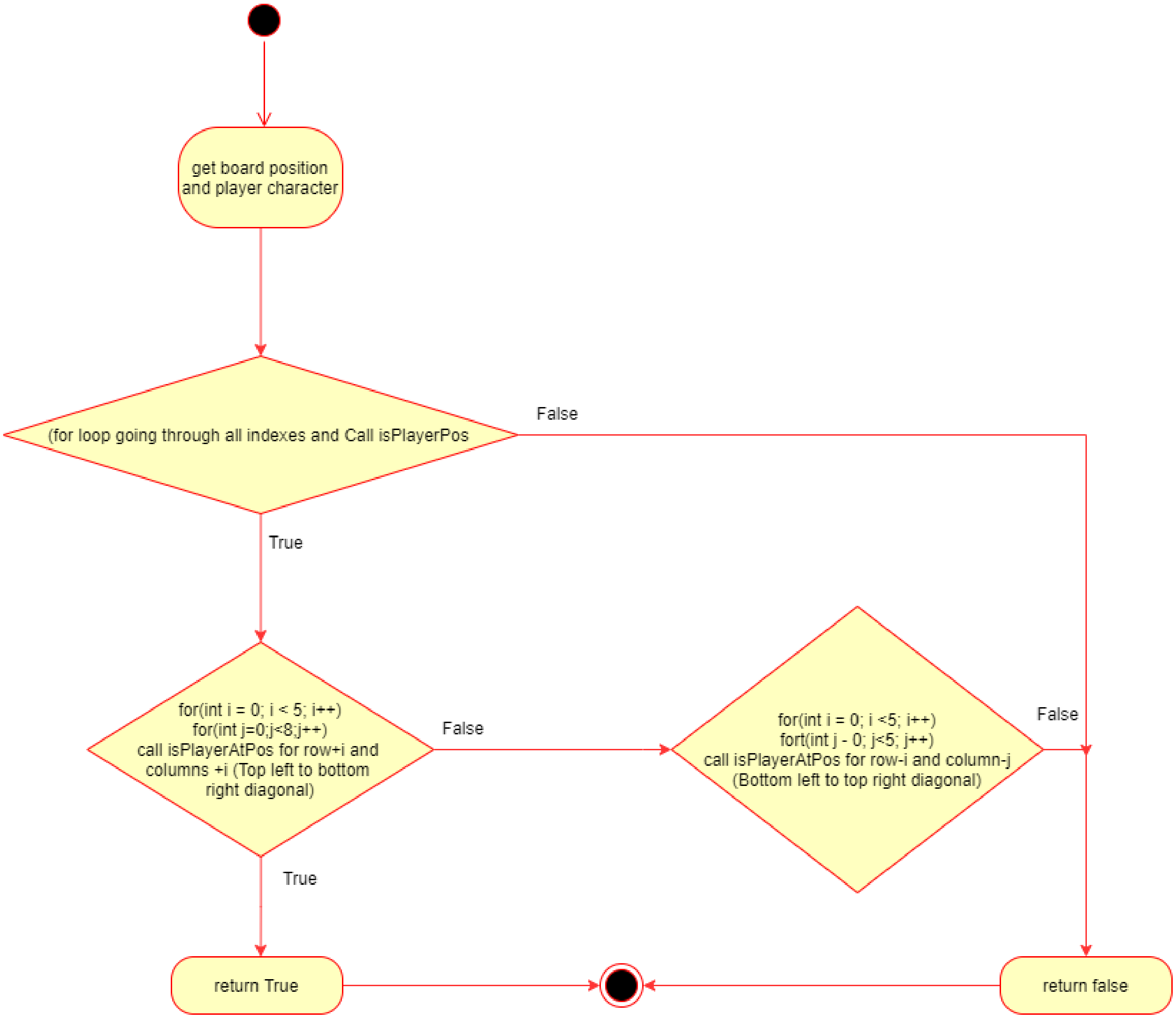
checkHorizontalWin


get board position and player character
for loop going through all the indexes and Call isPlayerAtPos
False
True
for(int i = 0; i < 5; i++)
call isPlayerAtPos for columns+i
False
True
return True
return false

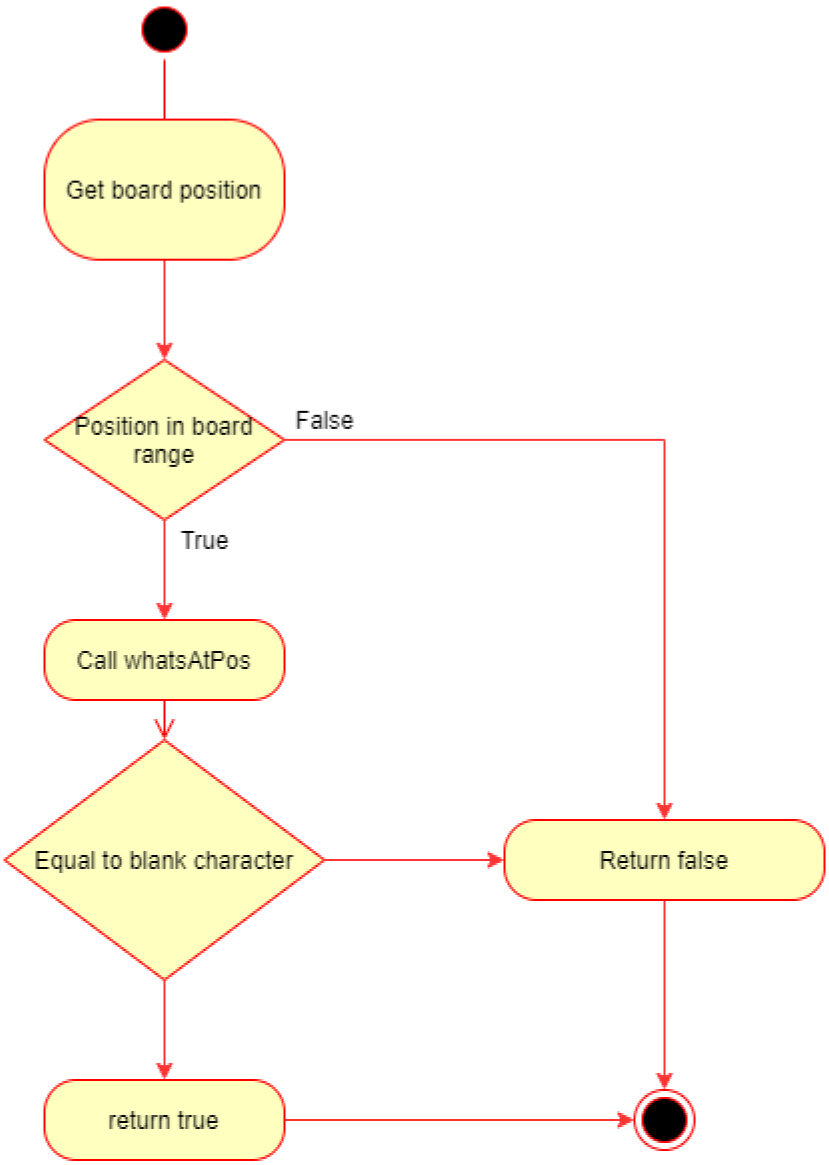
checkVerticalWin


get board position
and player character
for loop going through all indexes and Call isPlayerAtPos
False
True
for(int i = 0; i < 8; i++)
call isPlayerAtPos for row+i
False
True
return True
return false

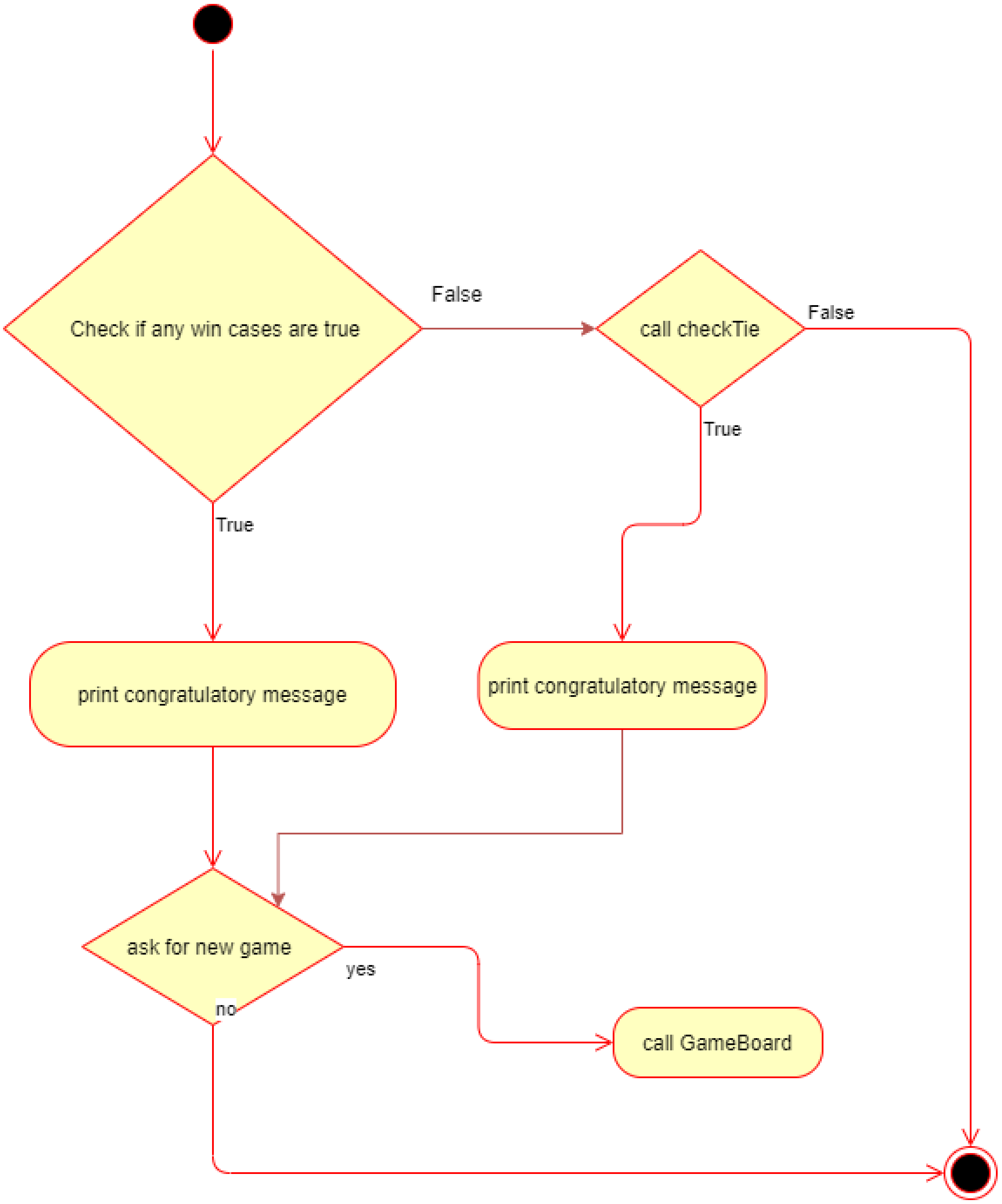
checkDiaganolWin


get board position and player character
(for loop going through all indexes and Call isPlayerPos
False
True
for(int i = 0; i < 5; i++)
for(int j=0;j<8;j++)
call isPlayerAtPos for row+i and columns +i (Top left to bottom right diagonal)
False
for(int i = 0; i <5; i++)
fort(int j - 0; j<5; j++)
call isPlayerAtPos for row-i and column-j
(Bottom left to top right diagonal)
False
True
return True
return false

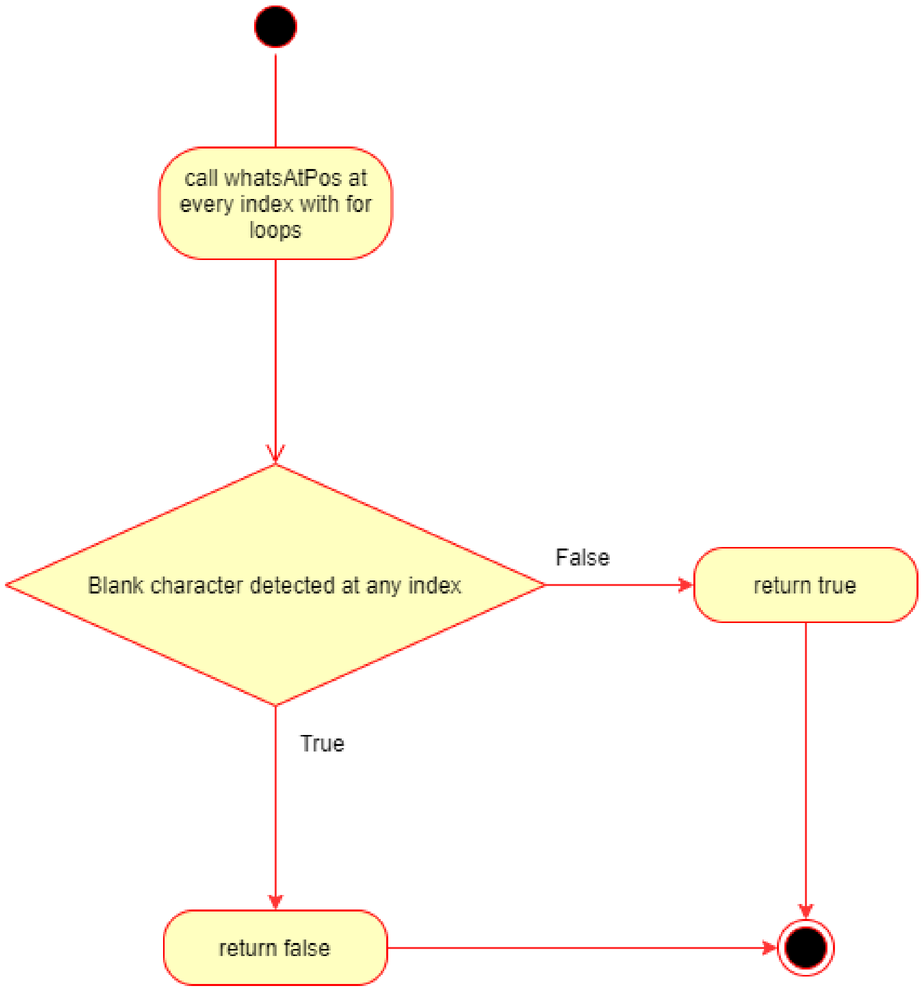
checkIfFree



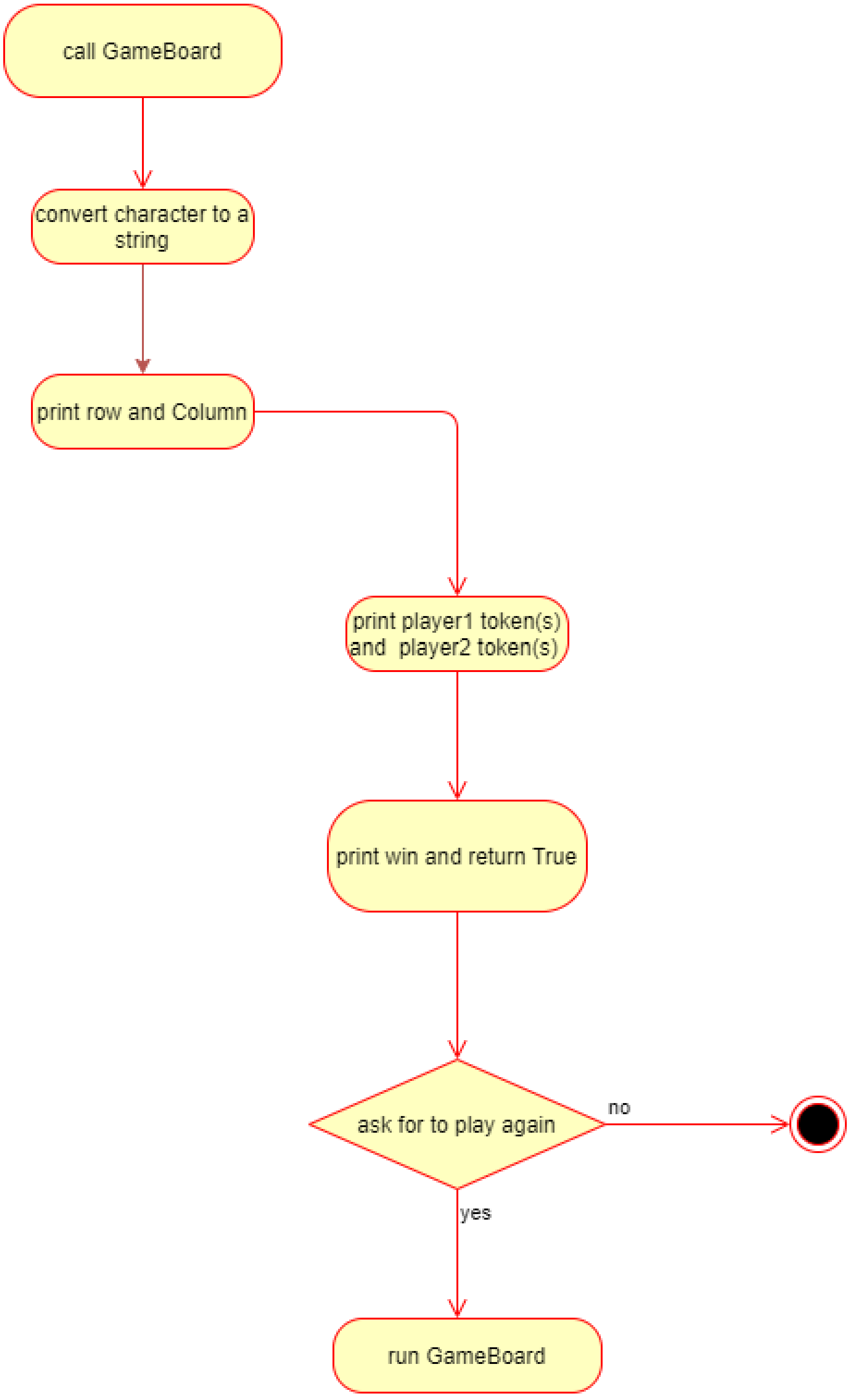
checkForWin


Check if any win cases are true
False
call checkTie
False
True
True
print congratulatory message
print congratulatory message
ask for new game
yes
no
call GameBoard

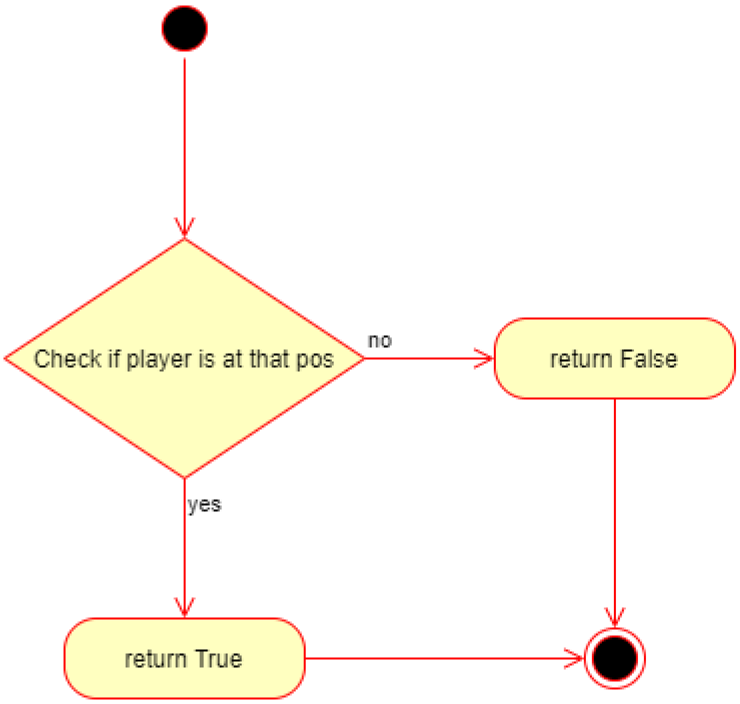
checkTie

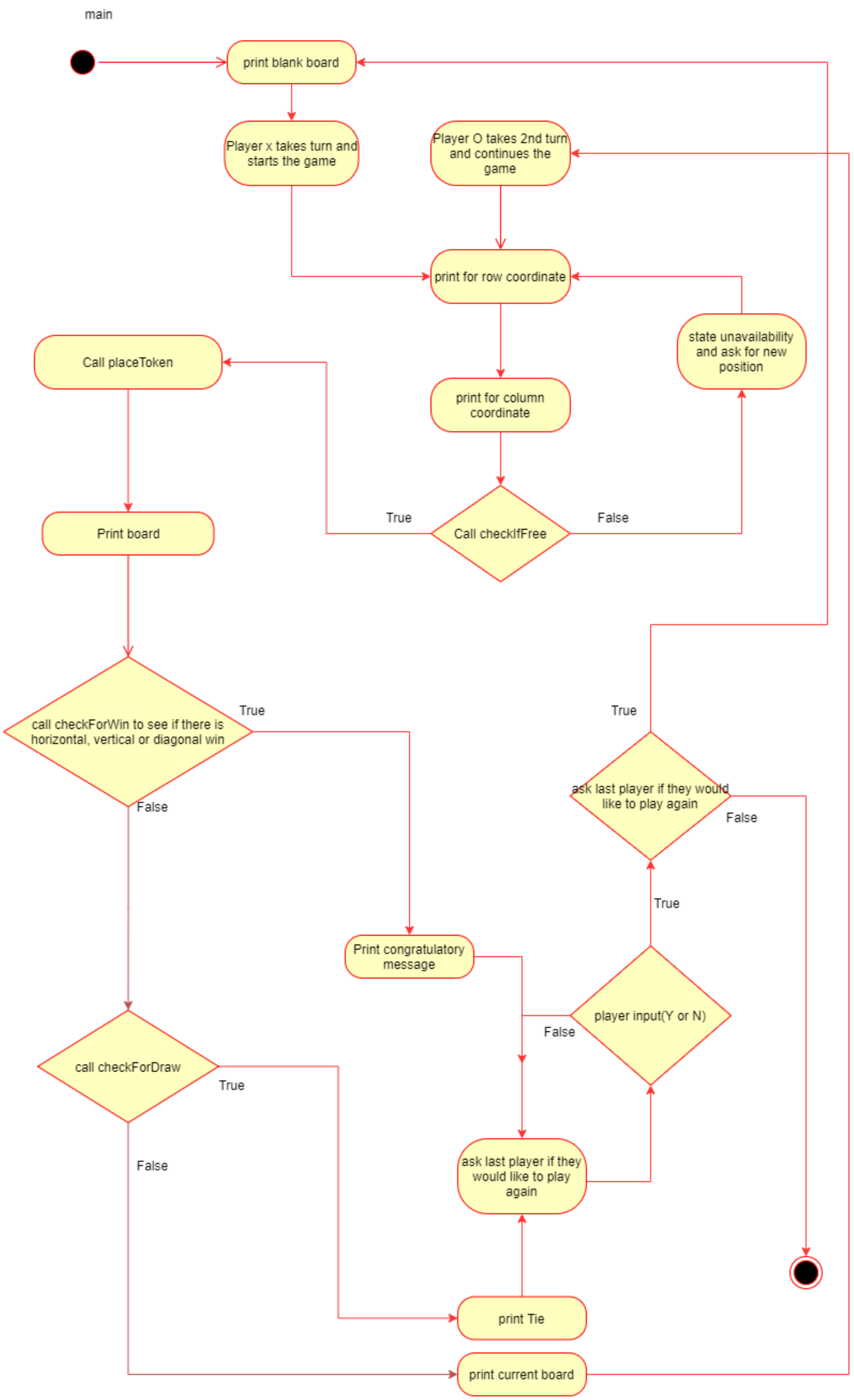

call whatsAtPos at every index with for loops
Blank character detected at any index
False
return true
True
return false

toString


call GameBoard
convert character to a string
print row and Column
print player1 token(s) and player2 token(s)
print win and return True
ask for to play again
no
yes
run GameBoard

isPlayerAtPos




main
print blank board
Player x takes turn and starts the game
Player O takes 2nd turn and continues the game
print for row coordinate
state unavailability and ask for new position
Call placeToken
print for column coordinate
Print board
True
Call checkIfFree
False
True
call checkForWin to see if there is horizontal, vertical or diagonal win
True
False
ask last player if they would like to play again
False
True
Print congratulatory message
player input(Y or N)
False
call checkForDraw
True
False
ask last player if they would like to play again
print Tie
print current board