CPSC 2150 Project Report

Carson Tollison

Requirements Analysis

Functional Requirements:

1. As a player I need be able to see the board so that I know which positions are available on the game board.
2. As a player I need to know which marker I use so that I know who's turn it is.
3. As a player I need to be able to input my column number so that I can put my marker into that position.
4. As a player I need to know how the columns are labeled so that I place my marker into the correct position.
5. As a player I need to be able to choose if I want to play again or not so that I can quit or continue playing.
6. As a player I can input my column again if I have invalid input so that I get my marker on the board.
7. As a player I need to know the game status so that I will know if it was a win, tie, or loss.
8. As a player, if I get 5 in a row horizontally, I will win the game so that I can win the game
9. As a player, if I get 5 in a row vertically, I will win the game so that I can win the game
10. As a player, if I get 5 in a row diagonally, I will win the game so that I can win the game
11. As a player, I can pick again if I pick a column that does not exist, so I don't lose my turn
12. As a player, I can make a move after my opponent does (assuming they don't win), so I can always have my turn
13. As a player, I can end the game in a tie by taking the last space on the board without getting 5 in a row, so the game can end

Non-Functional Requirements

1. The system must be programmed with java.
2. The system must run on Unix.
3. The system should be reliable.
4. The system should be fast.
5. Board is of size 6 x 9

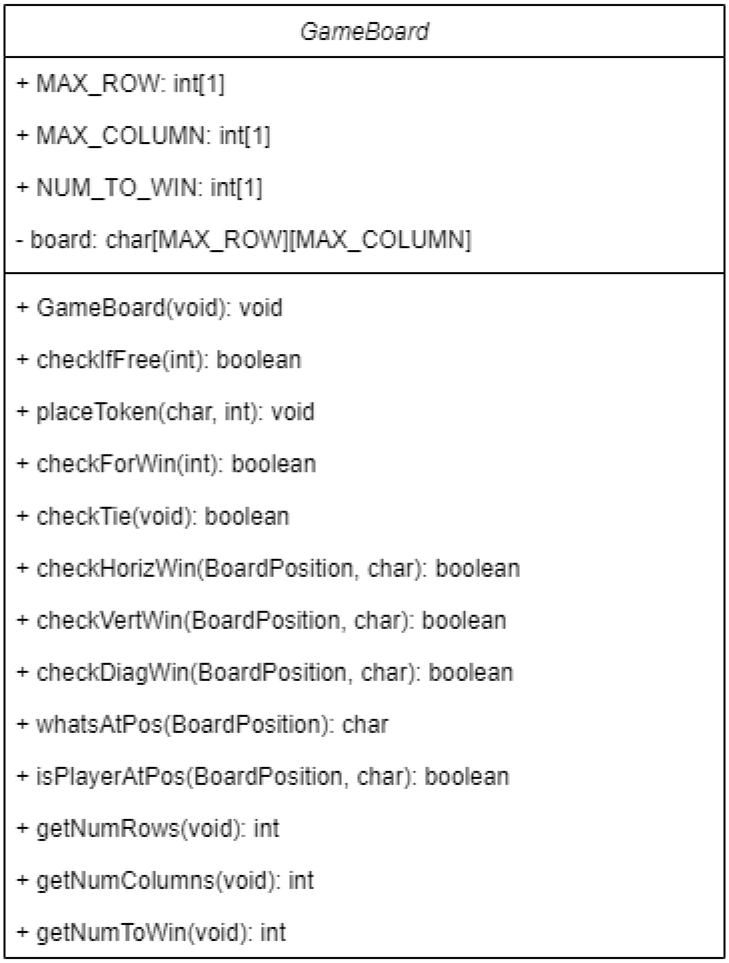
Deployment Instructions

default: compiles code. Runs with the *make* command.
run: runs code. Runs with the *make run* command.
clean: removes compiled (.class) files. Runs with the *make clean* command.

System Design

Class 1: GameBoard.java

Class diagram



GameBoard

+ MAX_ROW: int[1]

+ MAX_COLUMN: int[1]

+ NUM_TO_WIN: int[1]

- board: char[MAX_ROW][MAX_COLUMN]

+ GameBoard(void): void

+ checkIfFree(int): boolean

+ placeToken(char, int): void

+ checkForWin(int): boolean

+ checkTie(void): boolean

+ checkHorizWin(BoardPosition, char): boolean

+ checkVertWin(BoardPosition, char): boolean

+ checkDiagWin(BoardPosition, char): boolean

+ whatsAtPos(BoardPosition): char

+ isPlayerAtPos(BoardPosition, char): boolean

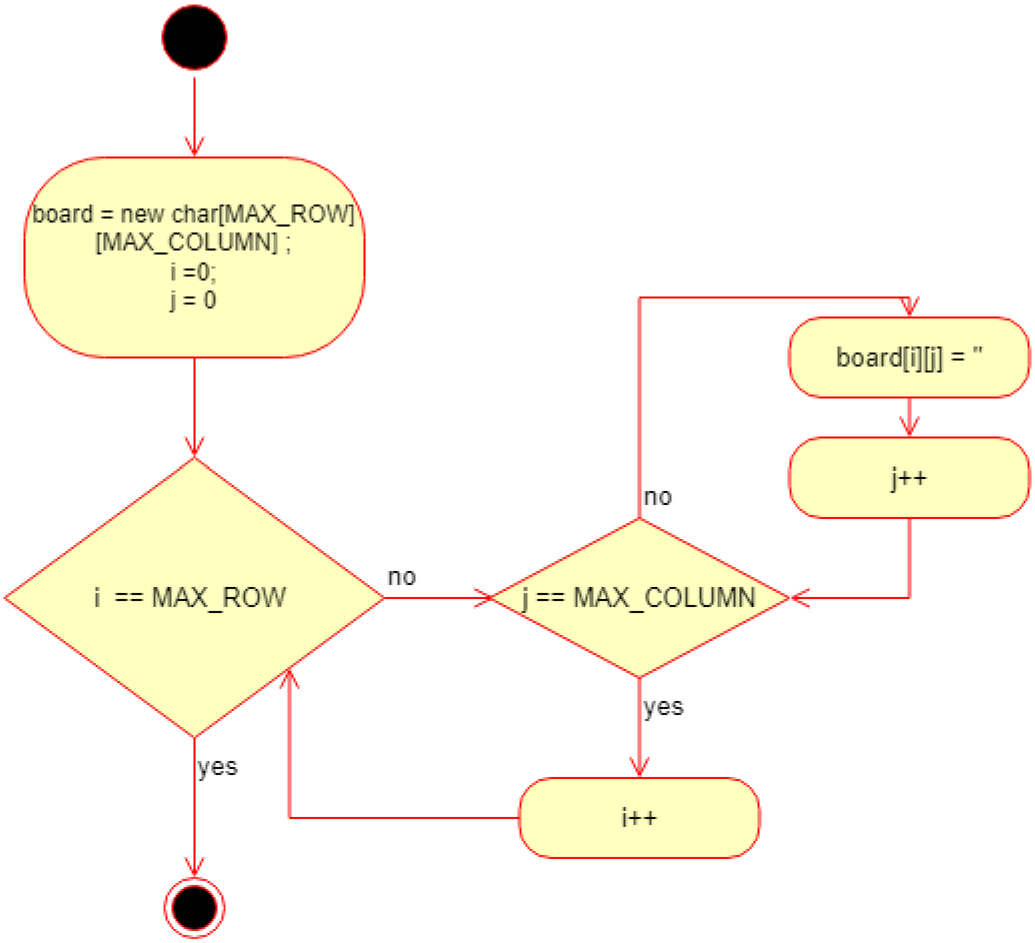
+ getNumRows(void): int

+ getNumColumns(void): int

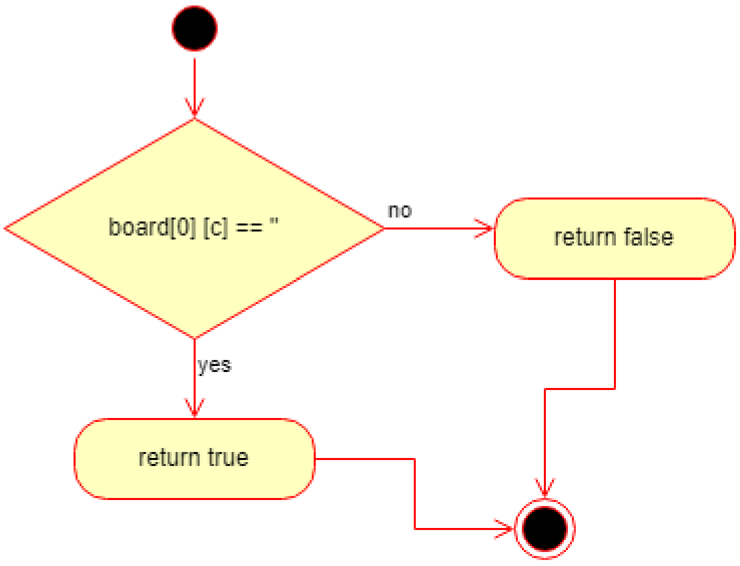
+ getNumToWin(void): int

Activity diagrams

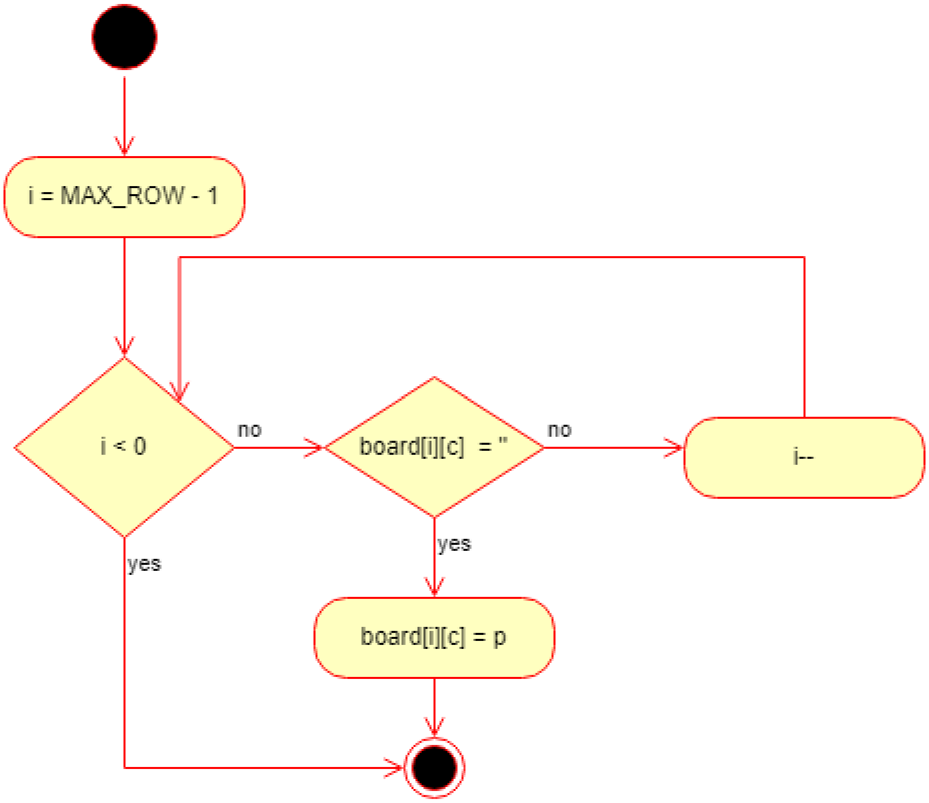
GameBoard



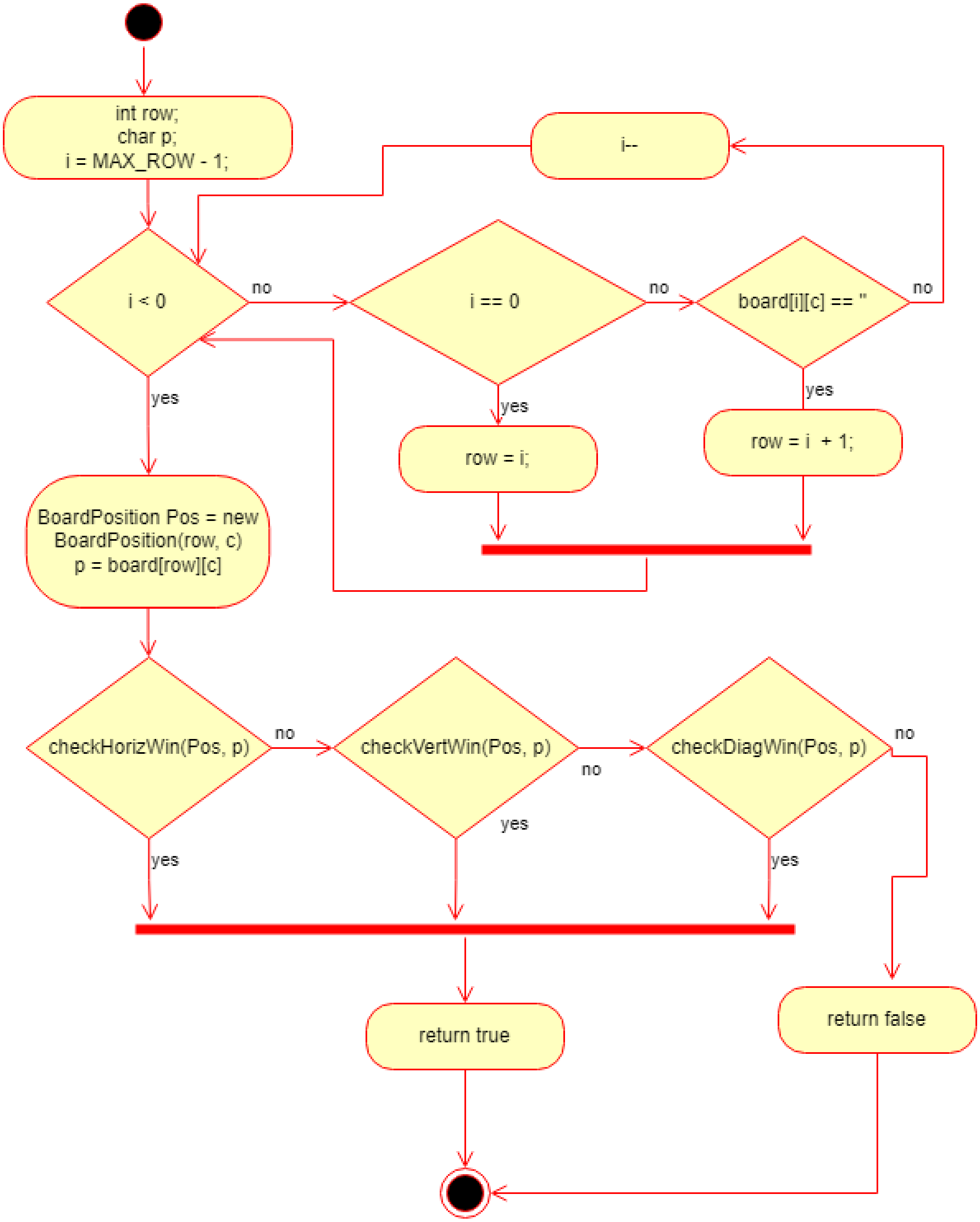
checkIfFree



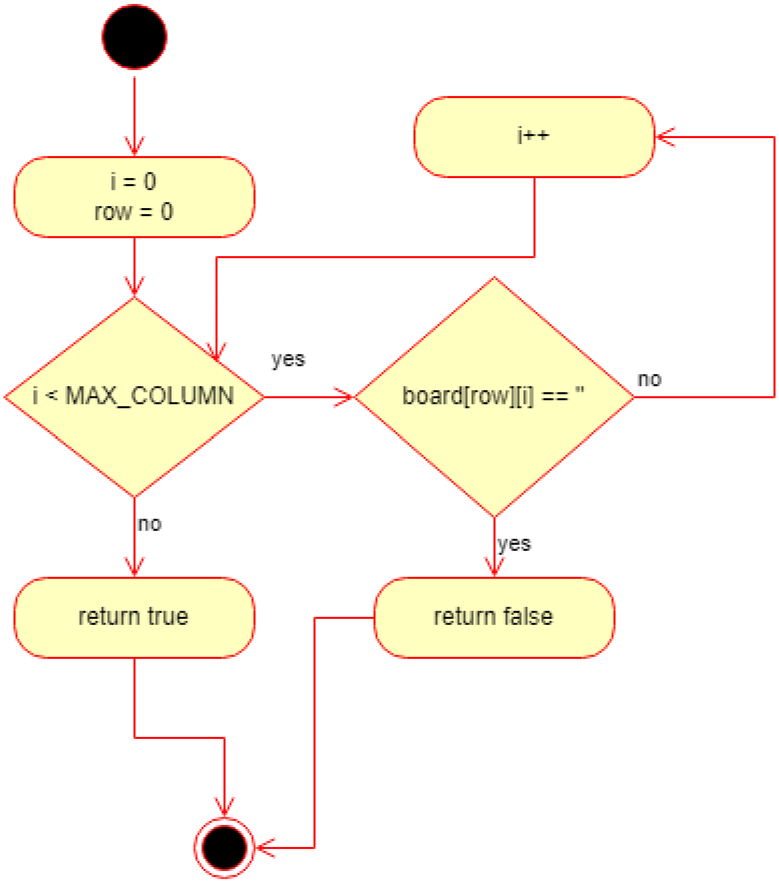
placeToken



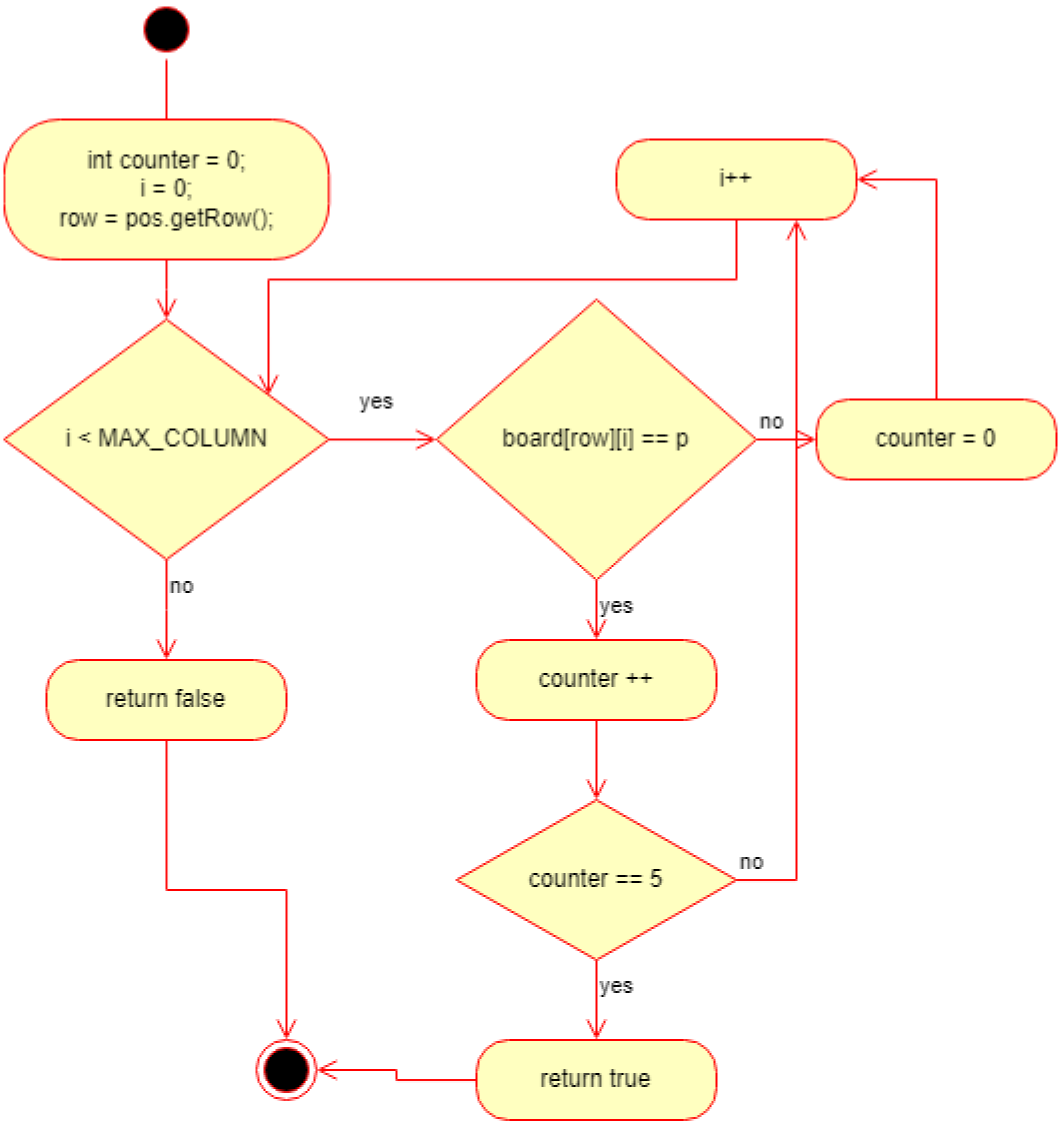
checkForWin



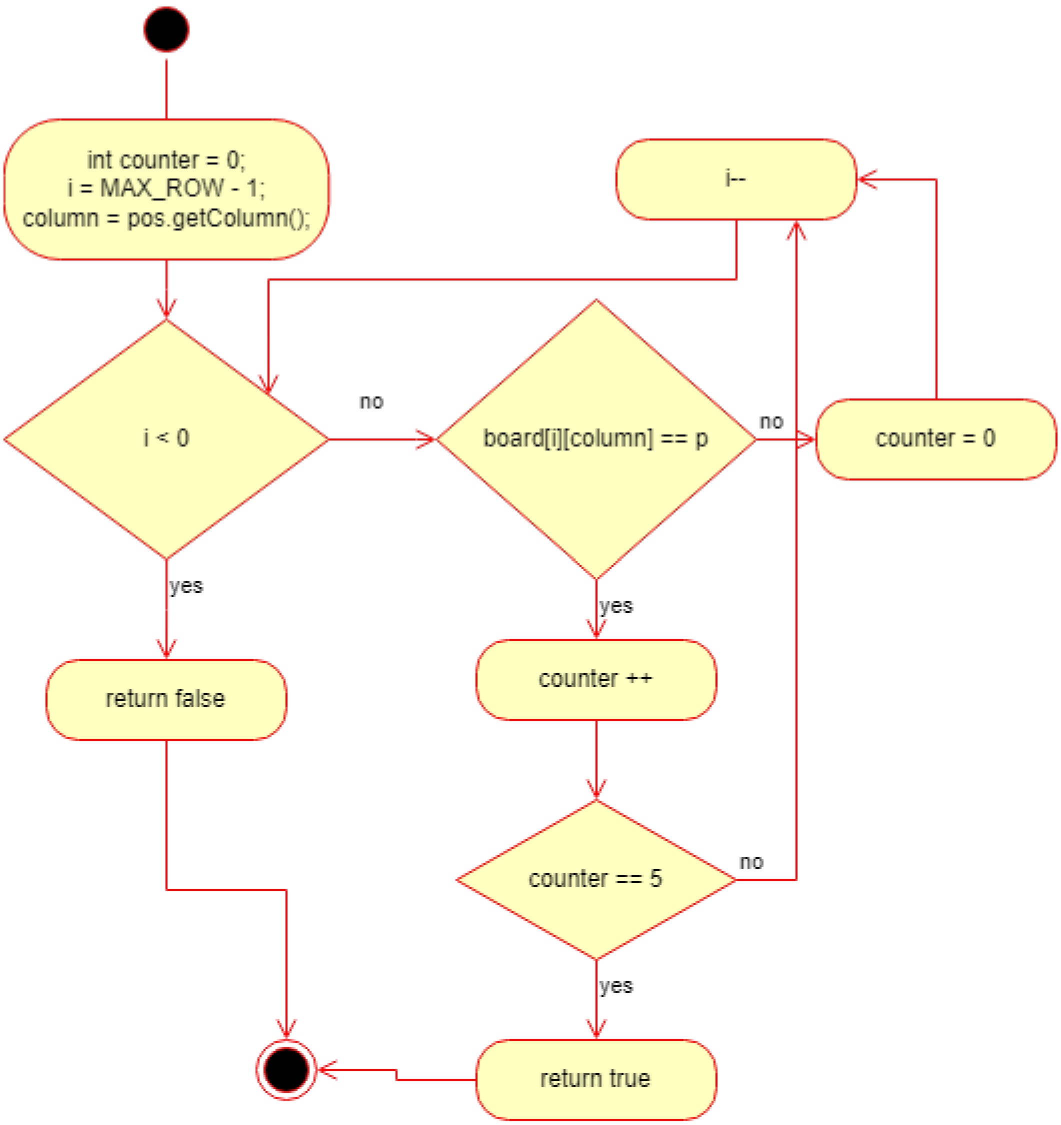
checkTie


i = 0
row = 0
i++
i < MAX_COLUMN
yes
board[row][i] == ''
no
no
yes
return true
return false

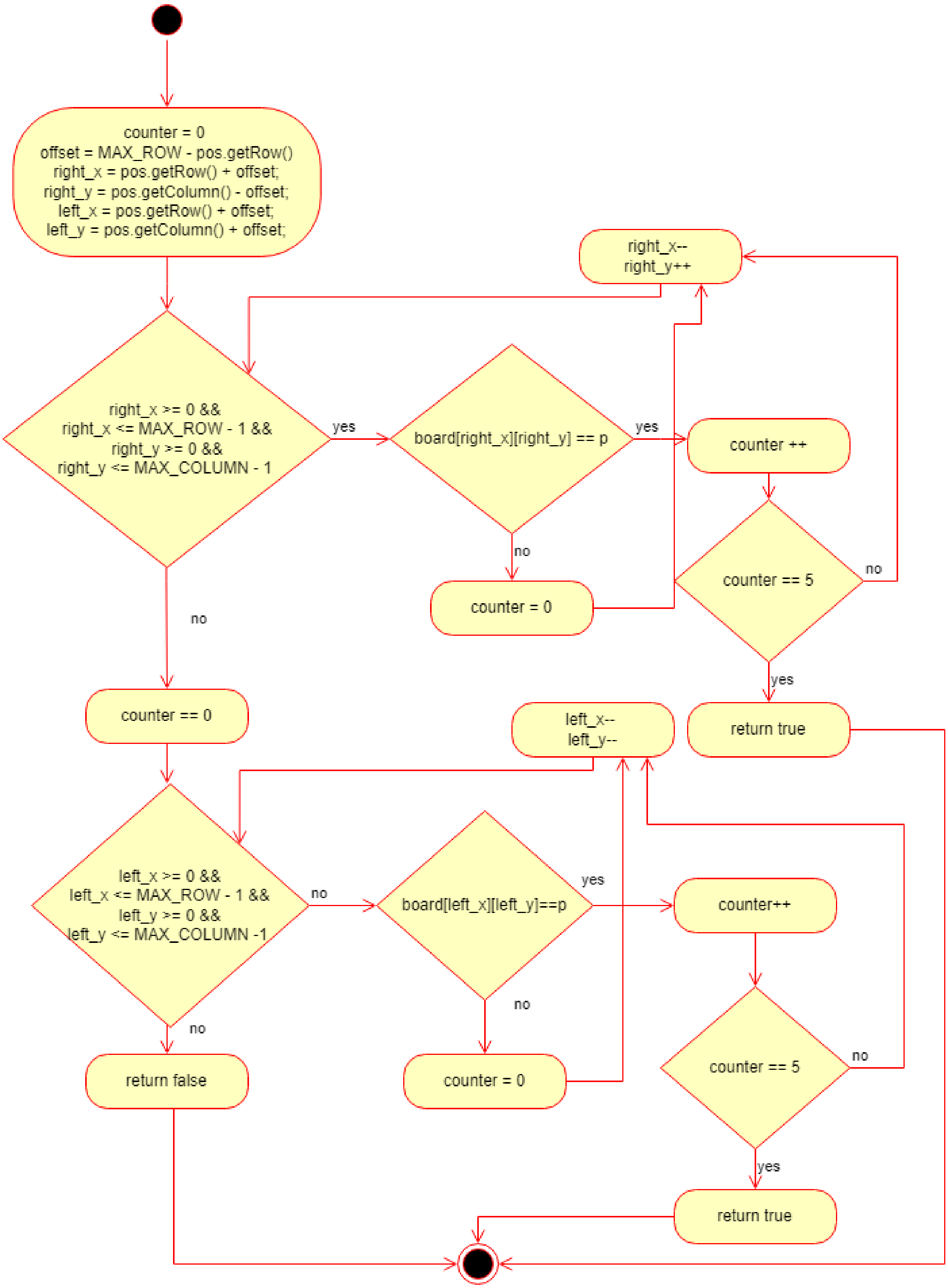
checkHorizWin



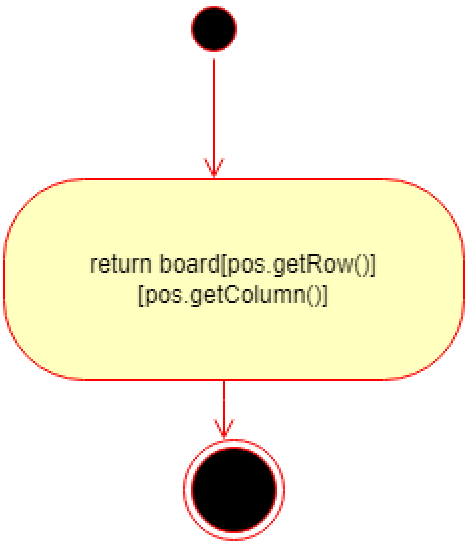
checkVertWin



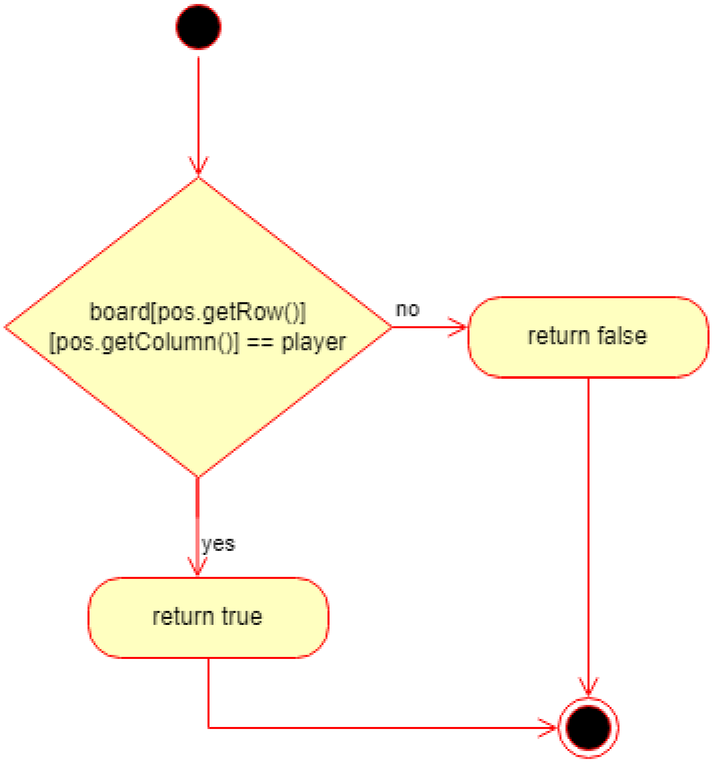
checkDiagWin



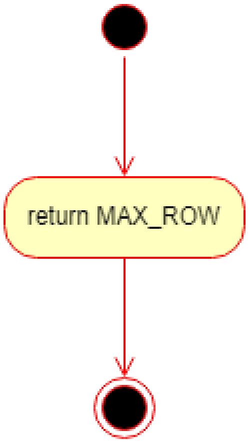
whatsAtPos



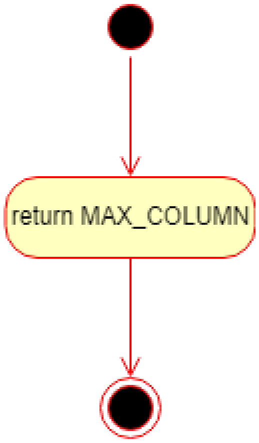
isPlayerAtPos



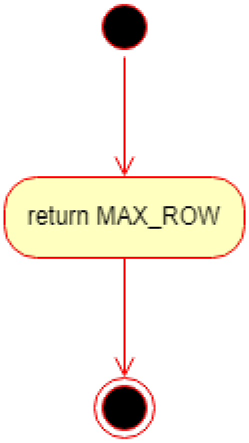
getNumRows



getNumColumns

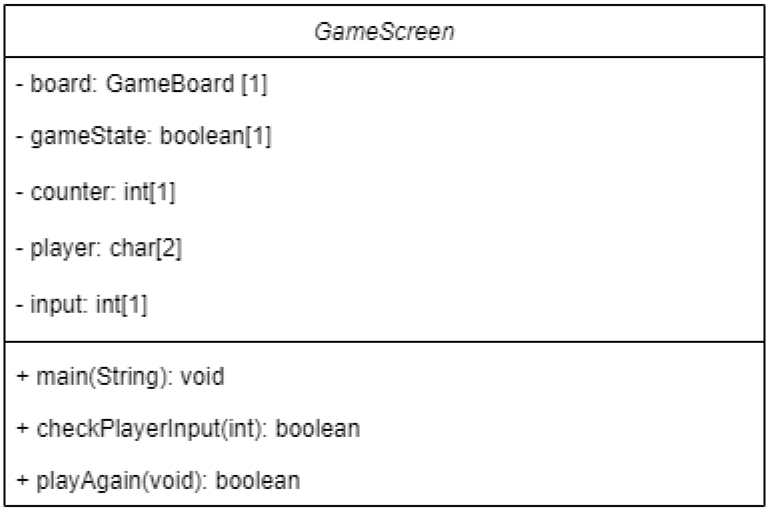


getNumToWin



Class 2: GameScreen.java

Class diagram



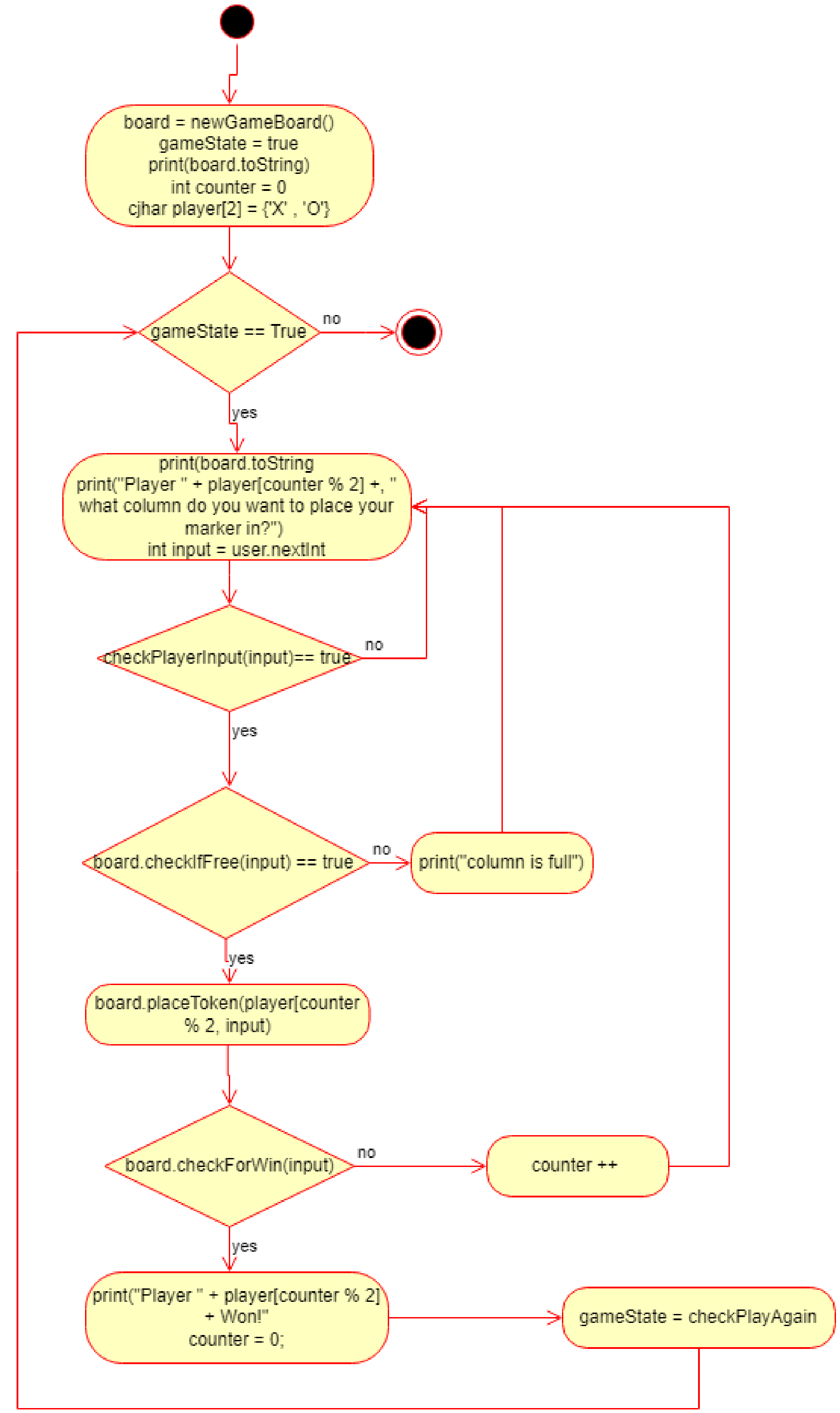
GameScreen

- board: GameBoard [1]
- gameState: boolean[1]
- counter: int[1]
- player: char[2]
- input: int[1]

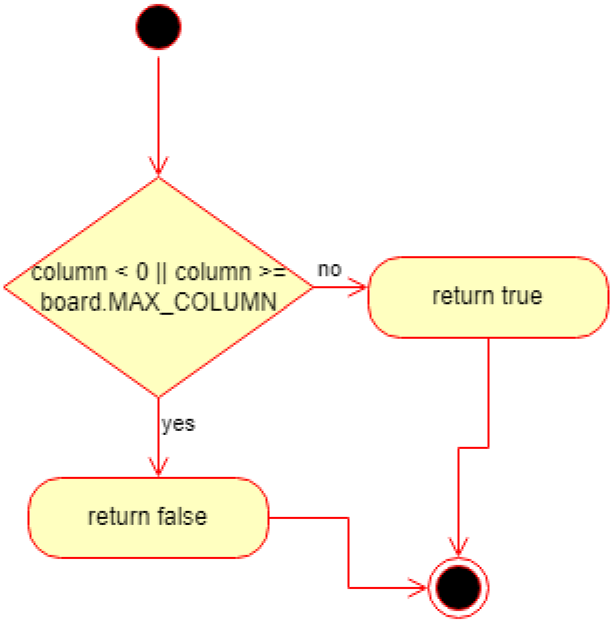
+ main(String): void
+ checkPlayerInput(int): boolean
+ playAgain(void): boolean

Activity diagrams

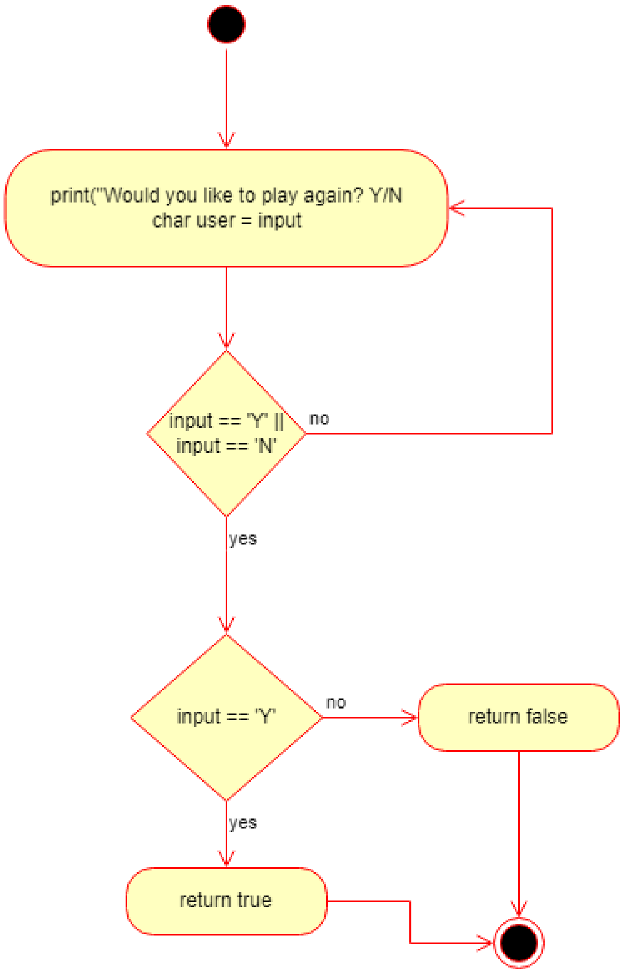
main


board = newGameBoard()
gameState = true
print(board.toString)
int counter = 0
cjhar player[2] = {'X' , 'O'}
gameState == True
no
yes
print(board.toString
print("Player " + player[counter % 2] +, " what column do you want to place your marker in?")
int input = user.nextInt
checkPlayerInput(input)== true
no
yes
board.checkIfFree(input) == true
no
print("column is full")
yes
board.placeToken(player[counter % 2, input)
board.checkForWin(input)
no
counter ++
yes
print("Player " + player[counter % 2] + Won!"
counter = 0;
gameState = checkPlayAgain

checkPlayerInput

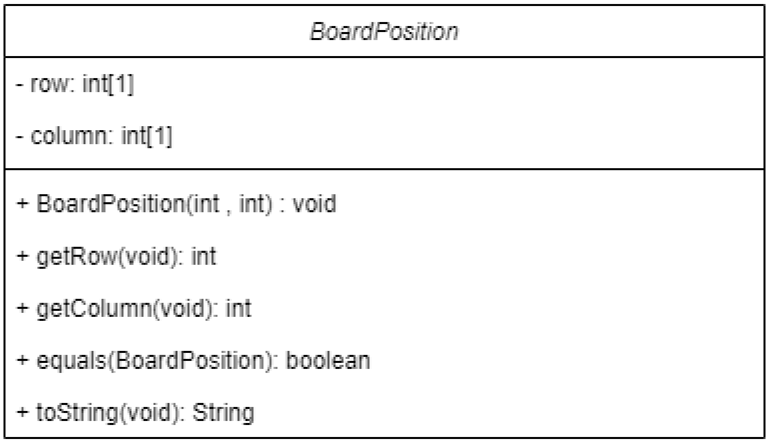


playAgain


print("Would you like to play again? Y/N
char user = input
input == 'Y' || input == 'N'
no
yes
input == 'Y'
no
return false
yes
return true

Class 3: BoardPosition.java

Class diagram



BoardPosition

- row: int[1]

- column: int[1]

+ BoardPosition(int , int) : void

+ getRow(void): int

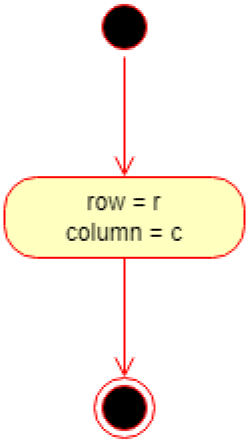
+ getColumn(void): int

+ equals(BoardPosition): boolean

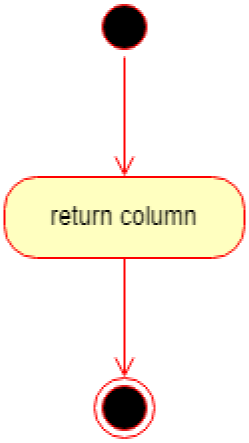
+ toString(void): String

Activity diagrams

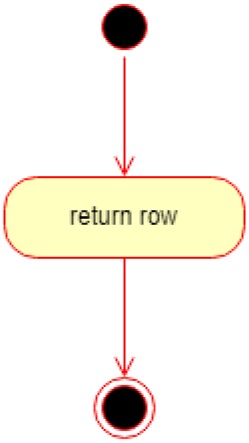
BoardPosition



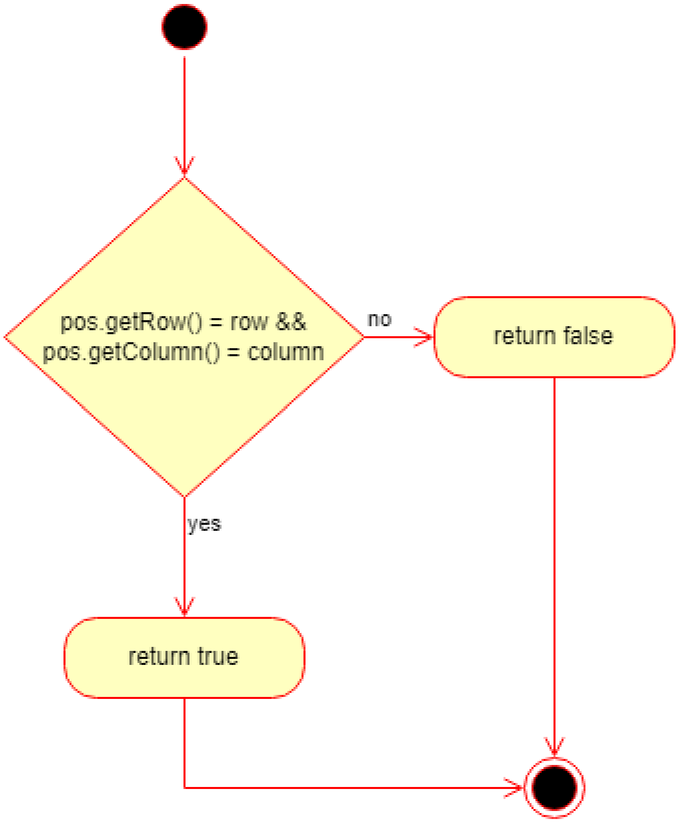
getRow



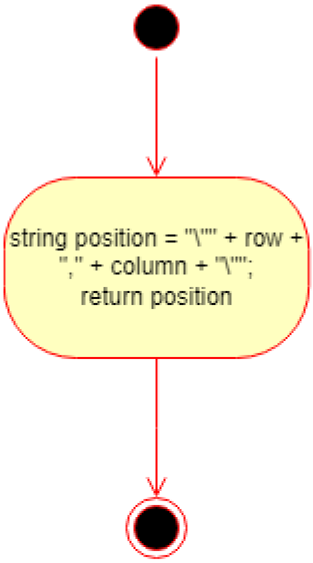
getColumn



equals



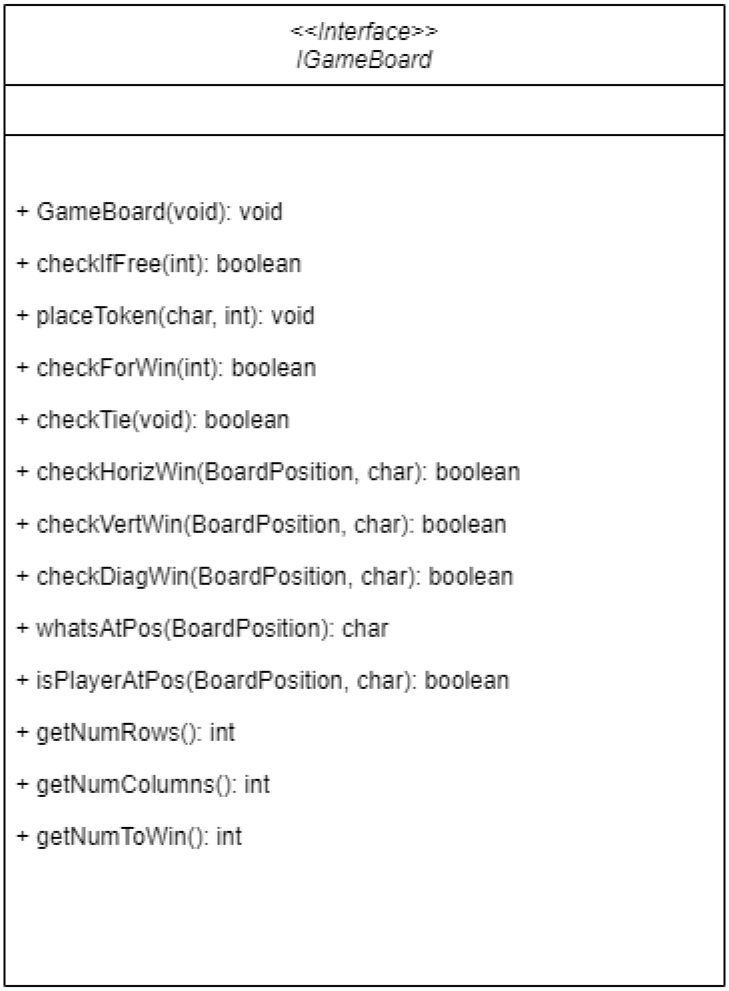
toString



```
string position = "\"" + row +
"," + column + "\"";
return position
```

Class 4: IGameBoard.java

Class diagram



<<Interface>>
IGameBoard

+ GameBoard(void): void

+ checkIfFree(int): boolean

+ placeToken(char, int): void

+ checkForWin(int): boolean

+ checkTie(void): boolean

+ checkHorizWin(BoardPosition, char): boolean

+ checkVertWin(BoardPosition, char): boolean

+ checkDiagWin(BoardPosition, char): boolean

+ whatsAtPos(BoardPosition): char

+ isPlayerAtPos(BoardPosition, char): boolean

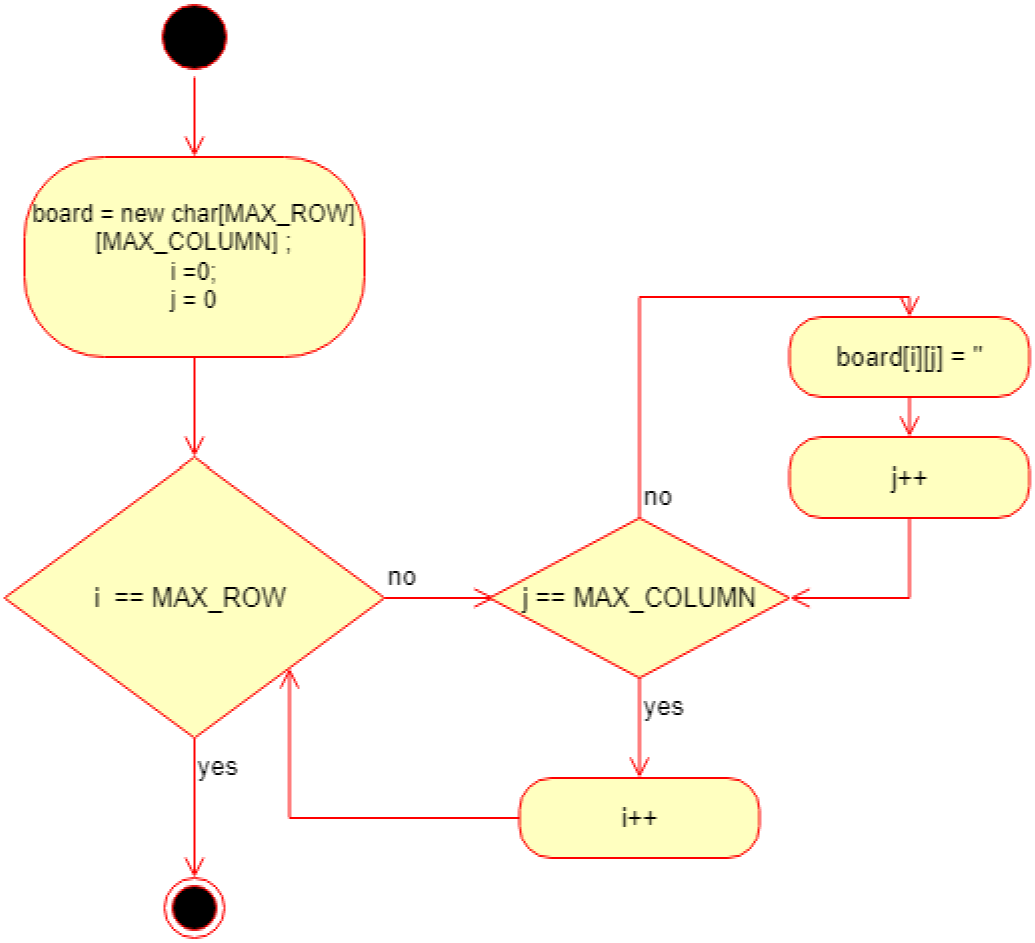
+ getNumRows(): int

+ getNumColumns(): int

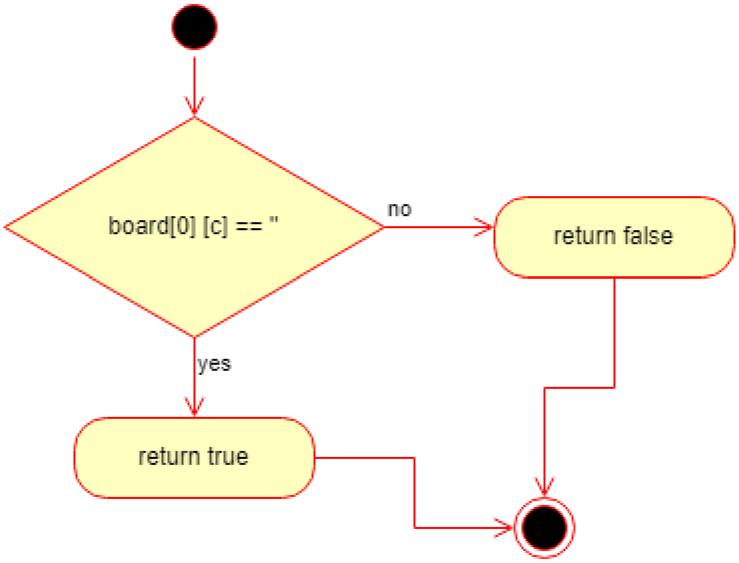
+ getNumToWin(): int

Activity diagrams

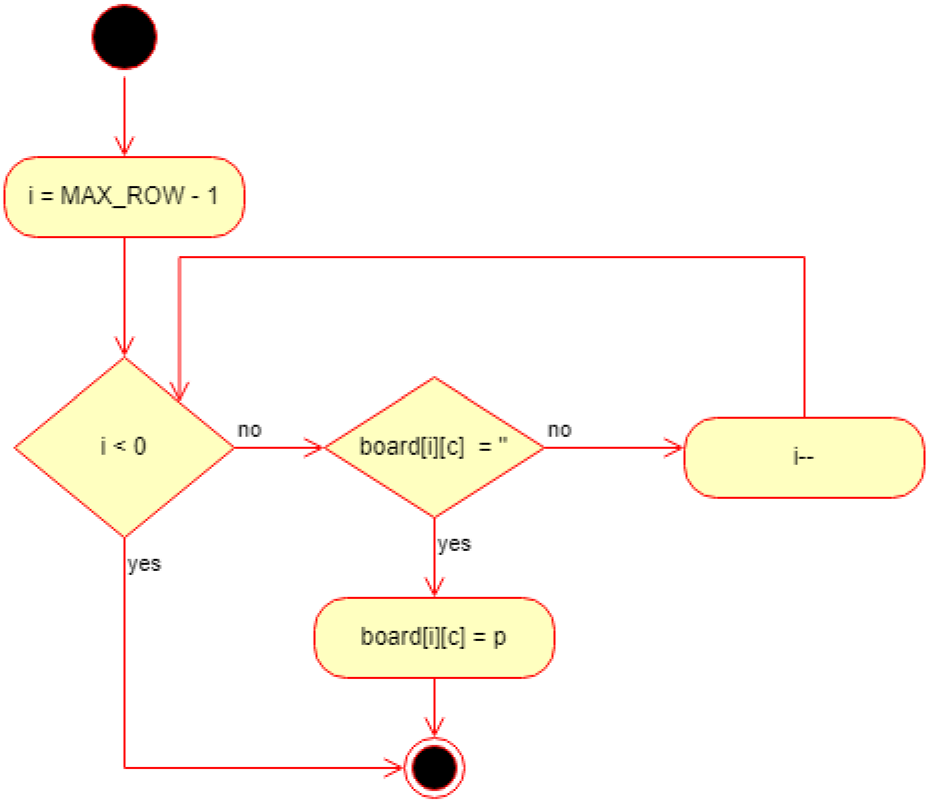
GameBoard



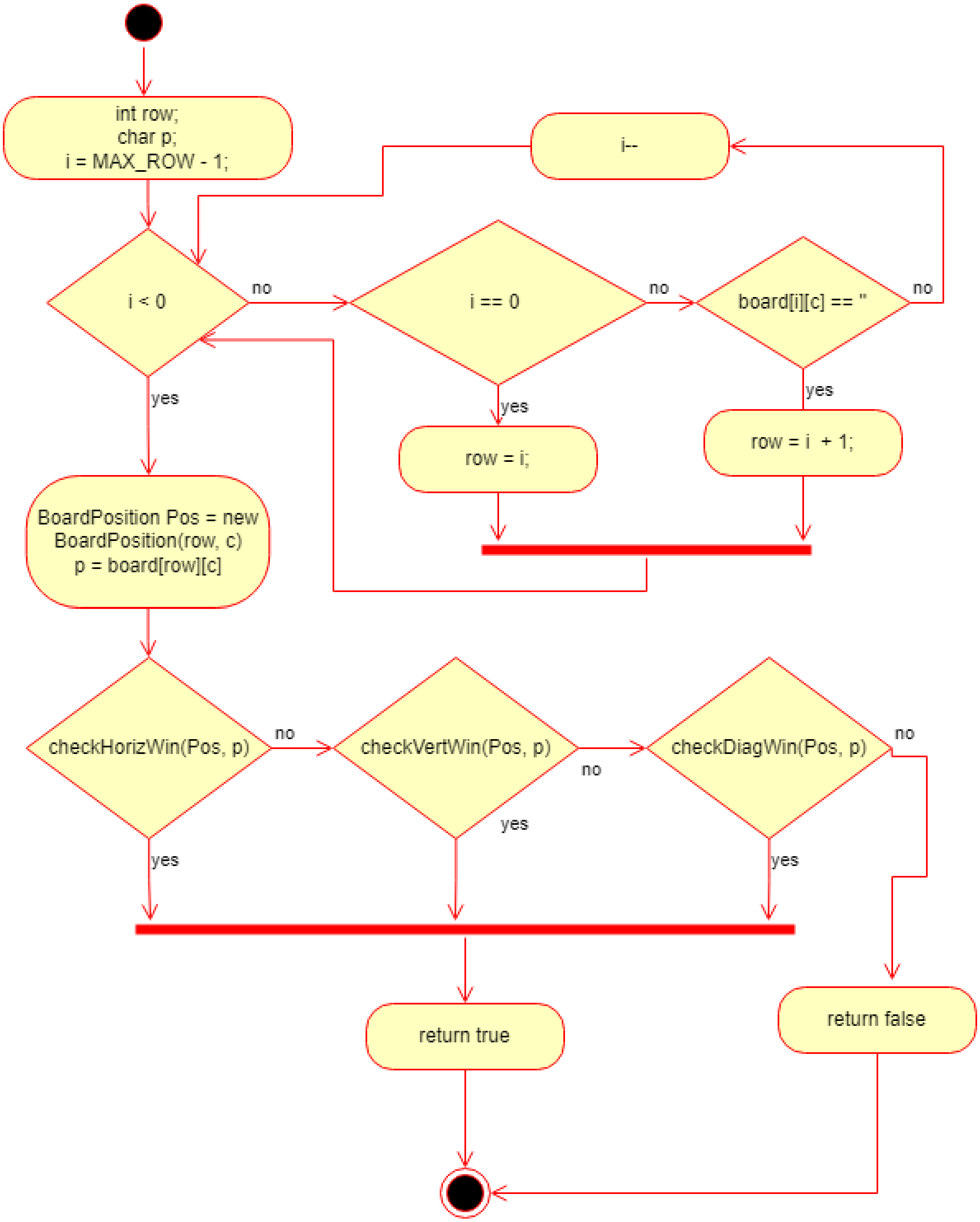
checkIfFree



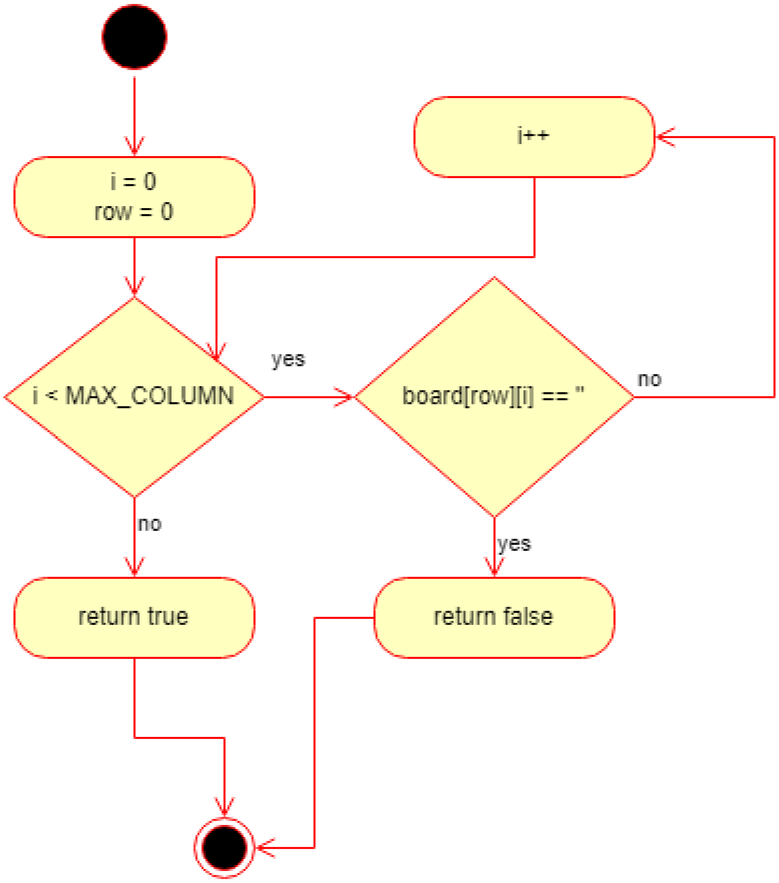
placeToken



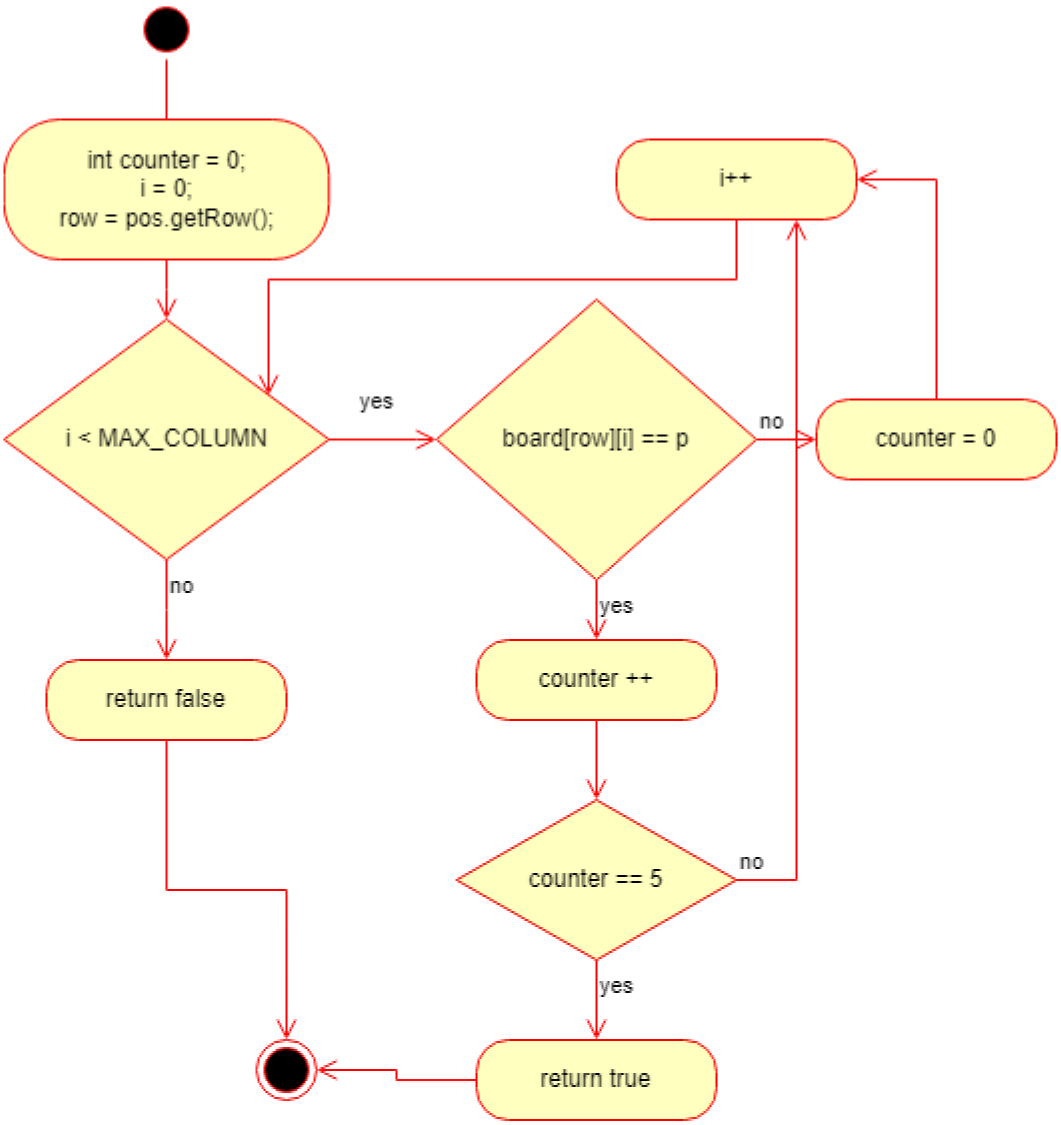
checkForWin



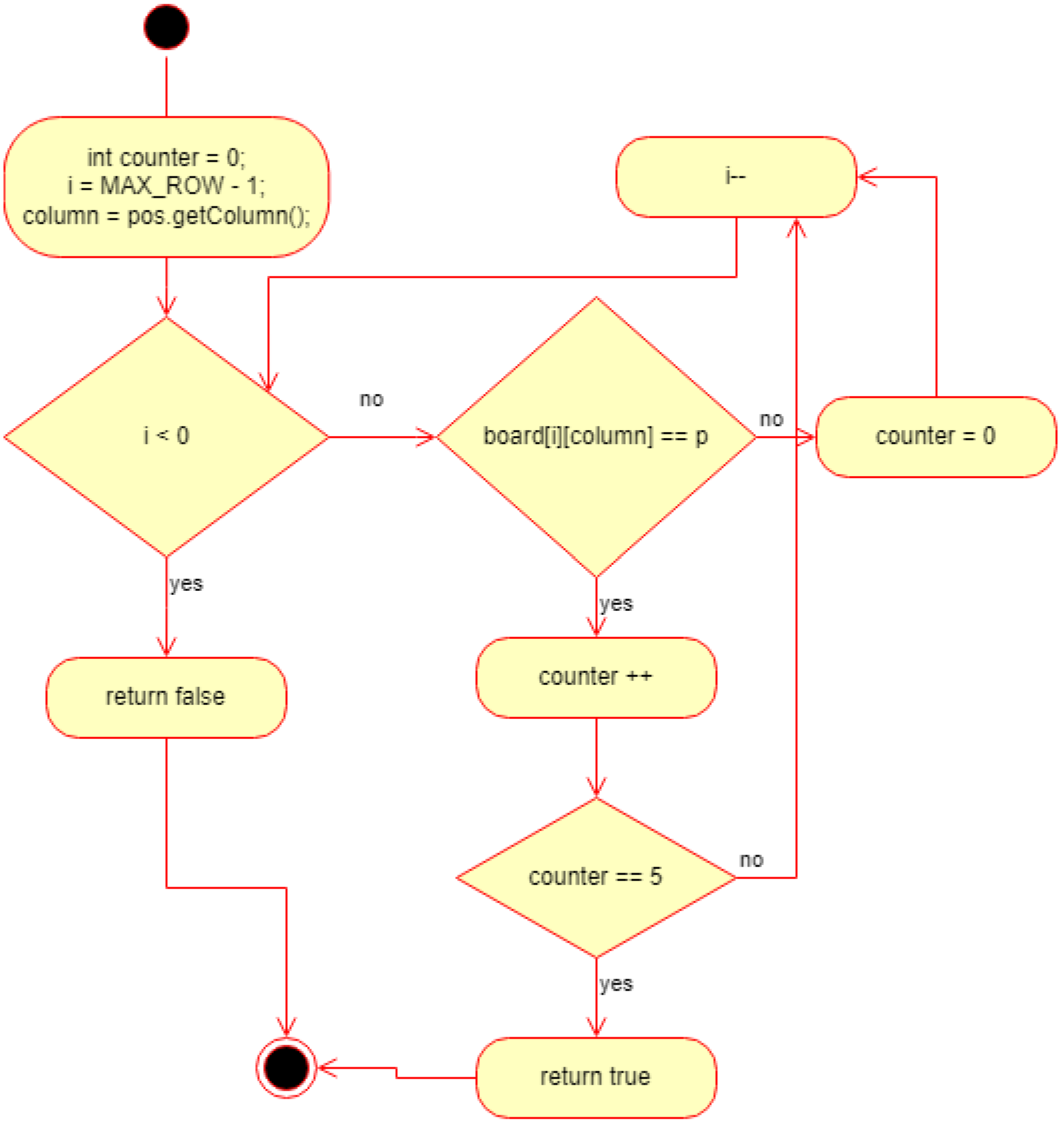
checkTie


i++
i = 0
row = 0
yes
i < MAX_COLUMN
board[row][i] == ''
no
no
yes
return true
return false

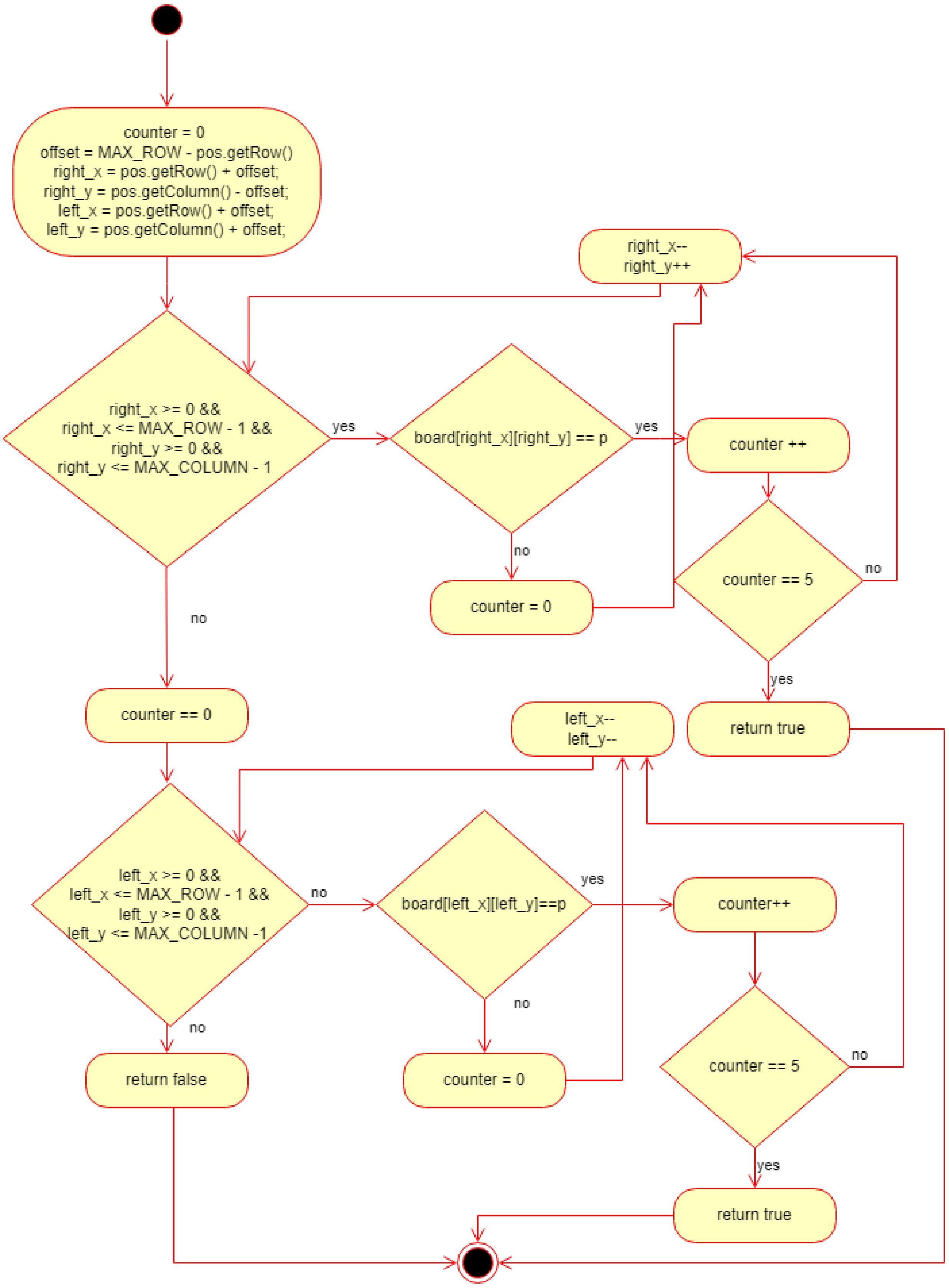
checkHorizWin



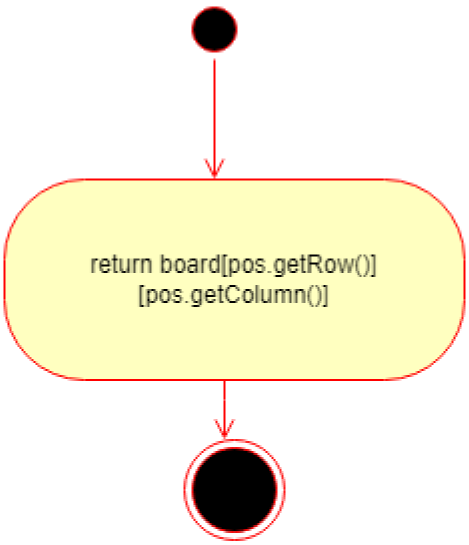
checkVertWin



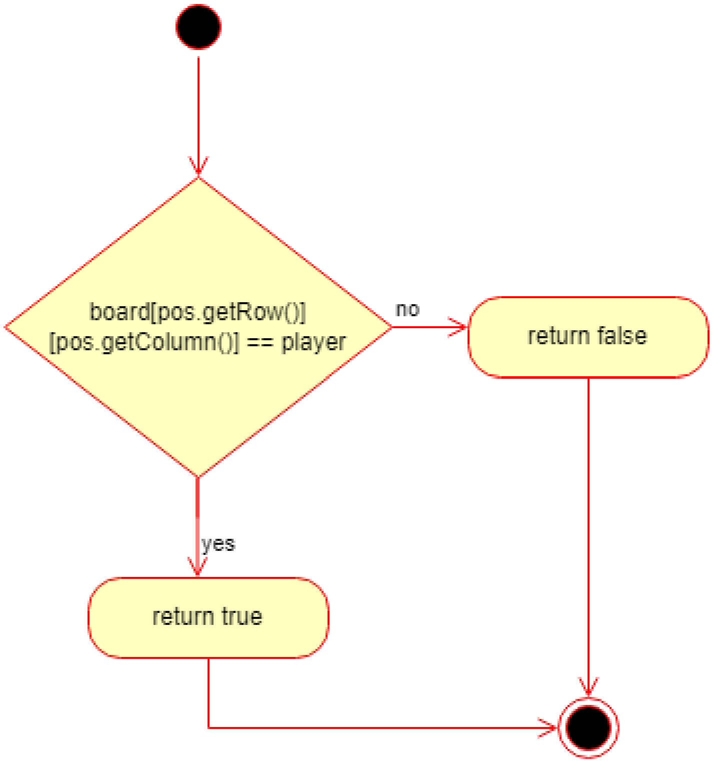
checkDiagWin



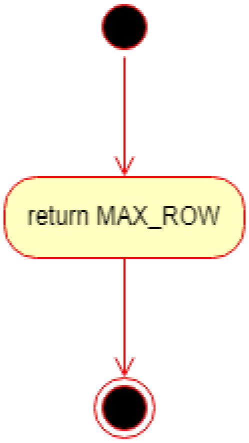
whatsAtPos



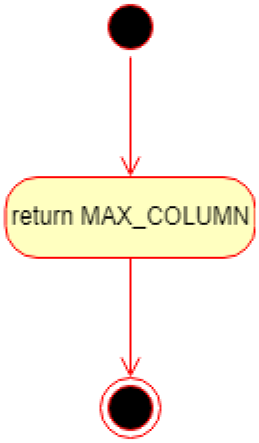
isPlayerAtPos



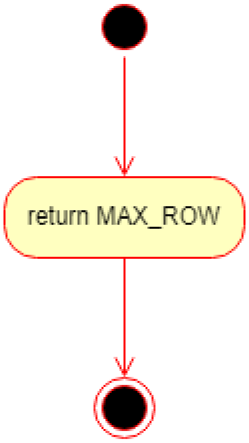
getNumRows



getNumColumns

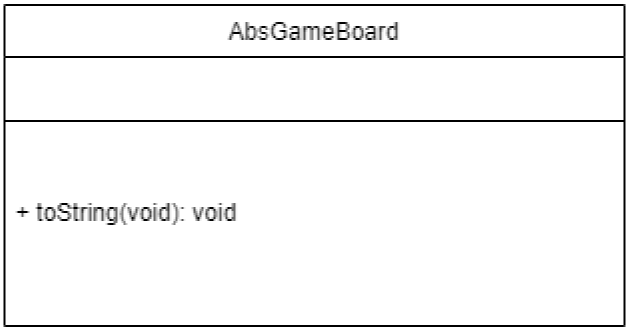


getNumToWin



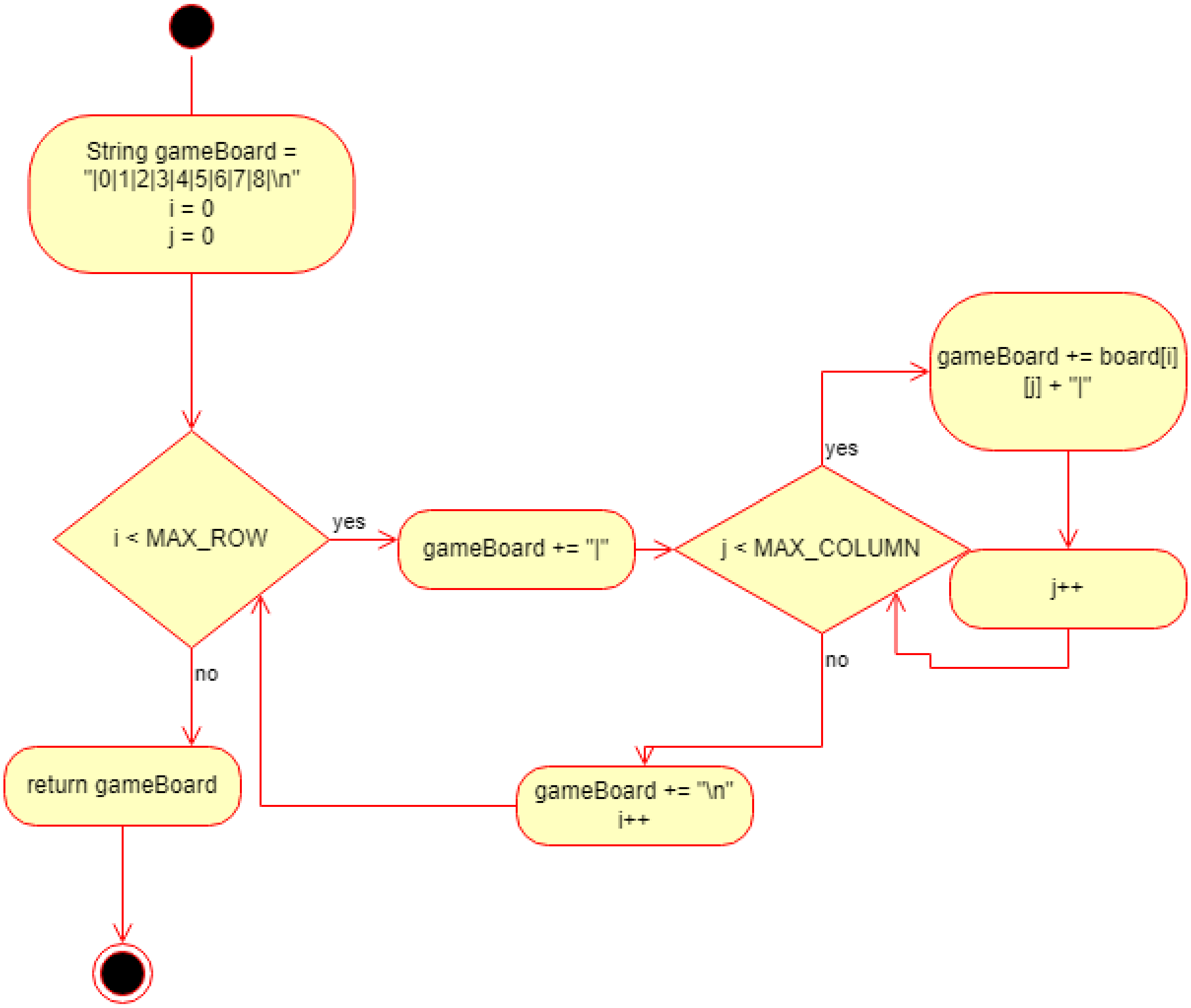
Class 5: AbsGameBoard.java

Class diagram



Activity diagrams

toString



Test Cases

Details in Project 4.