Manning Graham

Project 3

CPSC 2150

10-28-2021

Requirements Analysis

Functional Requirements:

1) Player

a) Place Token: As a player, I can decide where I wish to place my token during the game so I can play and try to win.

b) Exit Game: As a player, after a game is completed I have the option of whether or not to continue playing in a new game or exit the game.

c) Win game: As a player, after I place a piece, I am able to win 3 different ways. By having five pieces in a row horizontally, vertically, or diagonally.

d) Tie game: As a player, I can tie the game if there are no options of winning left for me or my opponent.

e) Out of bounds: As a player, if I choose a placement for a token out of bounds I am able to choose another placement to continue playing without error.

f) Playing again: As a player, after a game is completed I have the option of playing again.

g) Moving after opponent: As a player after my opponent has made their move, It is then my turn and I am able to make my move.

h) Winning Vertically: As a player, I am able to see if I have 5 of the same tokens stacked on top of one another so I can see If I have won the game

I) Winning Horizontally: As a player, I am able to see if I have 5 of the same tokens side by side of one another so I can see If I have won the game

J) Winning Diagonally: As a player, I am able to see if I have 5 of the same tokens stacked diagonally on top of one another, from either left of right, so I can see If I have won the game

e) Board Position Taken: As a player, if I choose a placement for a token that is already taken by another token I am able to choose another placement to continue playing without error.

f) Character: As a player, I can choose which character I would like to play as so I can tell where my positions are, in contrast to my opponent.

g) Number of rows: As I player I can choose the number of rows I would like to play on.

h) Number of columns: As a player, I can choose the number of columns I would like to play on.

I) Number of Markers: As a player, I can choose how many markers it takes in a row to win the game.

J) Character change: As a player, after a game has finished, I have the opportunity to change the character I am playing as if I would like.

K) Number of players: As a player I can choose how many players are going to be playing the game.

Non-Functional Requirements:

1) The program should know whether the player's column input location is valid or not.

2) The program should know whether the column input is even on the board.

3) The program should know whether the player has won after a move vertically.

4) The program should know if there are any more spaces left to be played or if it is a tie.

5) The program should know who the winner is and display that they are the winner immediately after the winning move.

6) The program should know to display a blank game board after a win if the users wish to play again.

7) The program should be able to tell what player is taking up space in a certain position.

8) The program should be able to tell what marker is in what position

9) The program should be able to display the current game board after each move.

11) The program should know 0,0 is the top left of the board.

12) The program should know that Player one goes first.

13) The program should be written in java.

14) The program should run on Unix.

15) The program should establish the Max number of players as 10 and the minimum 2.

16) The Program should establish the maximum number of rows and columns as 100 and minimum is 3.

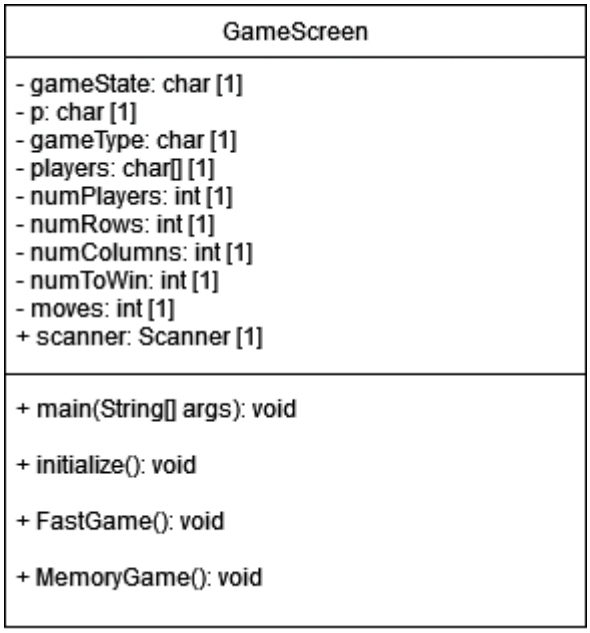
17) The program should establish the maximum number in a row to win as 25 and minimum is 3

3) The program should know whether the player has won after a move Horizontally.

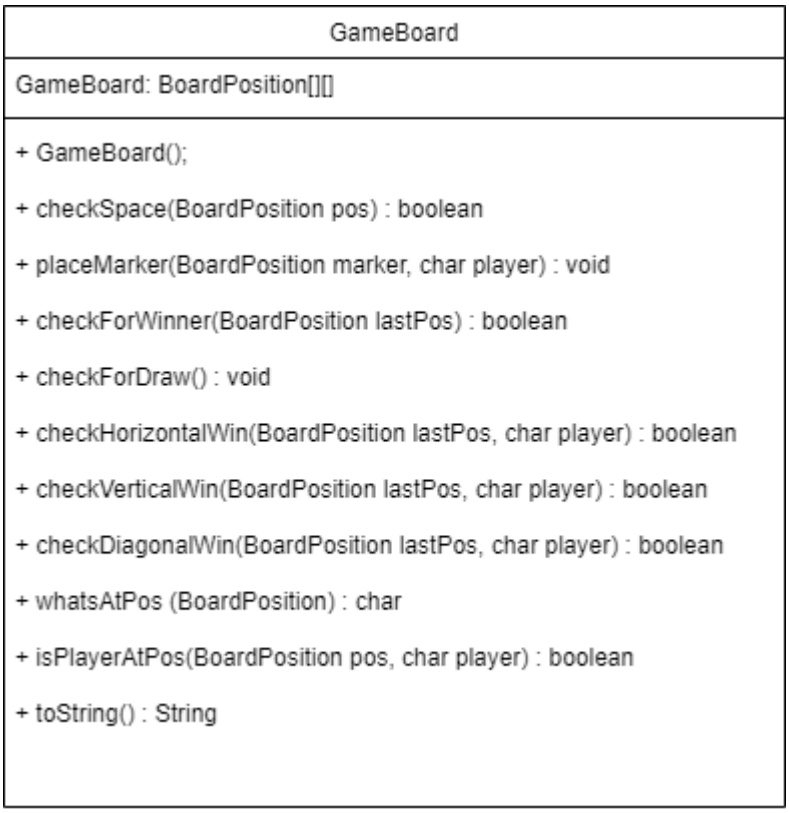
3) The program should know whether the player has won after a move Diagonally.

Class Diagrams

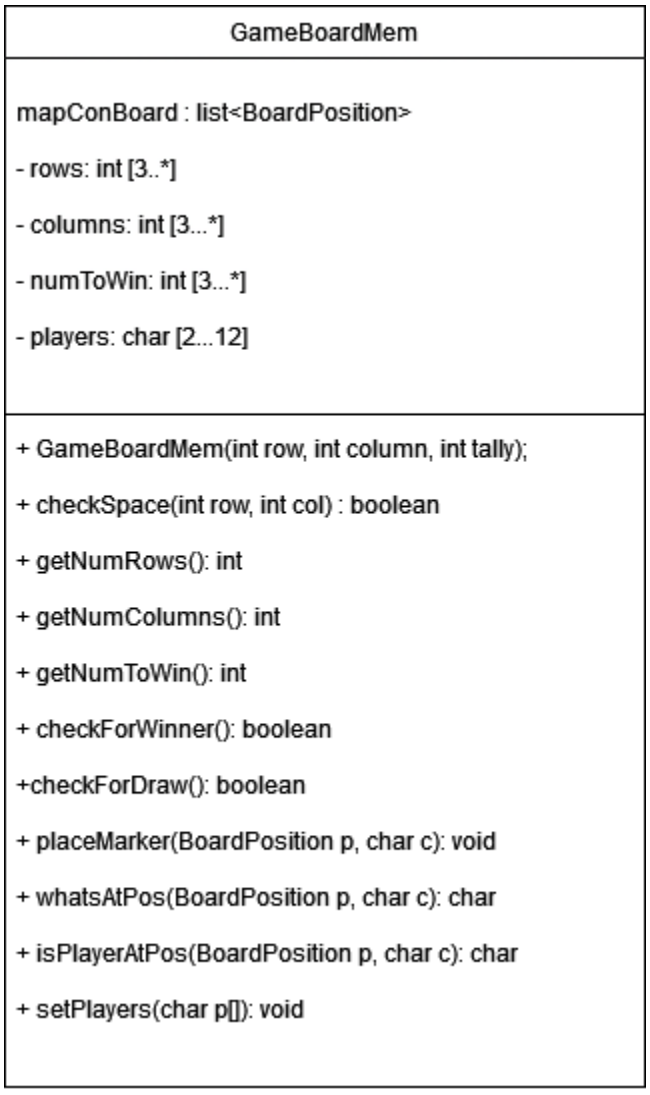
GameScreen:



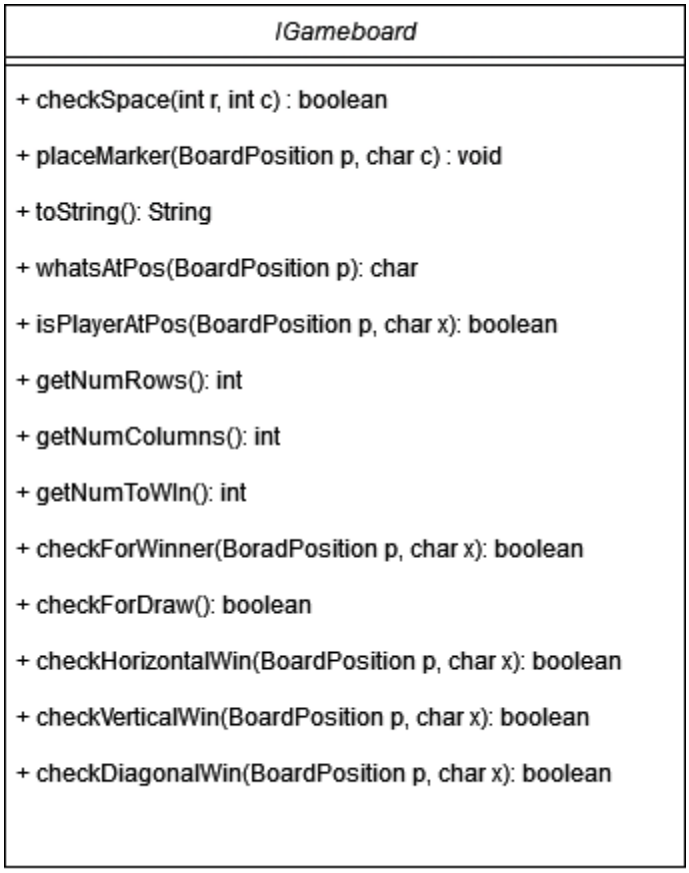
GameBoard:


GameBoard
GameBoard: BoardPosition[][]
+ GameBoard();
+ checkSpace(BoardPosition pos) : boolean
+ placeMarker(BoardPosition marker, char player) : void
+ checkForWinner(BoardPosition lastPos) : boolean
+ checkForDraw() : void
+ checkHorizontalWin(BoardPosition lastPos, char player) : boolean
+ checkVerticalWin(BoardPosition lastPos, char player) : boolean
+ checkDiagonalWin(BoardPosition lastPos, char player) : boolean
+ whatsAtPos (BoardPosition) : char
+ isPlayerAtPos(BoardPosition pos, char player) : boolean
+ toString() : String

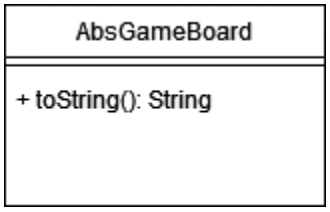
GameBoardMem:



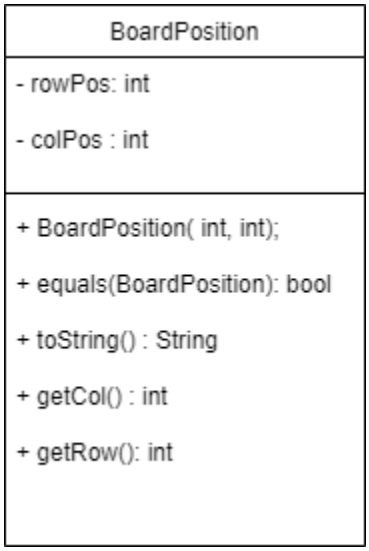
IGameBoard:


IGameboard
+ checkSpace(int r, int c) : boolean
+ placeMarker(BoardPosition p, char c) : void
+ toString(): String
+ whatsAtPos(BoardPosition p): char
+ isPlayerAtPos(BoardPosition p, char x): boolean
+ getNumRows(): int
+ getNumColumns(): int
+ getNumToWin(): int
+ checkForWinner(BoradPosition p, char x): boolean
+ checkForDraw(): boolean
+ checkHorizontalWin(BoardPosition p, char x): boolean
+ checkVerticalWin(BoardPosition p, char x): boolean
+ checkDiagonalWin(BoardPosition p, char x): boolean

AbsGameBoard:



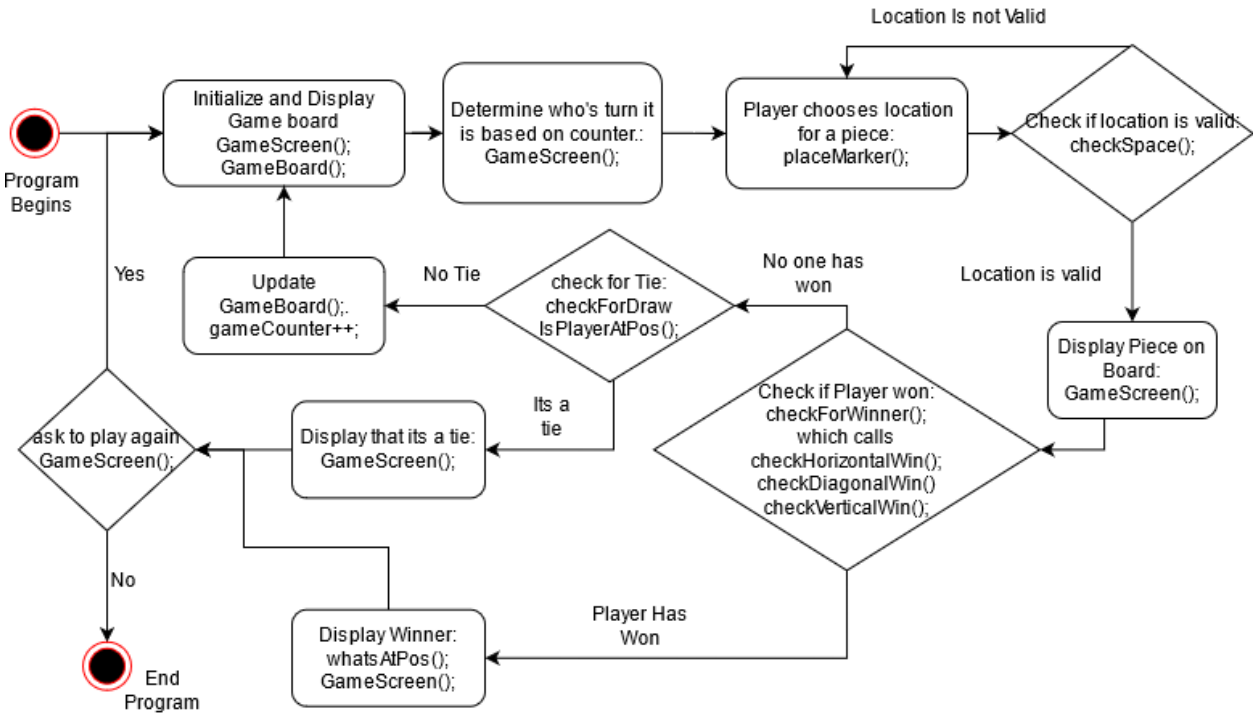
BoardPosition:



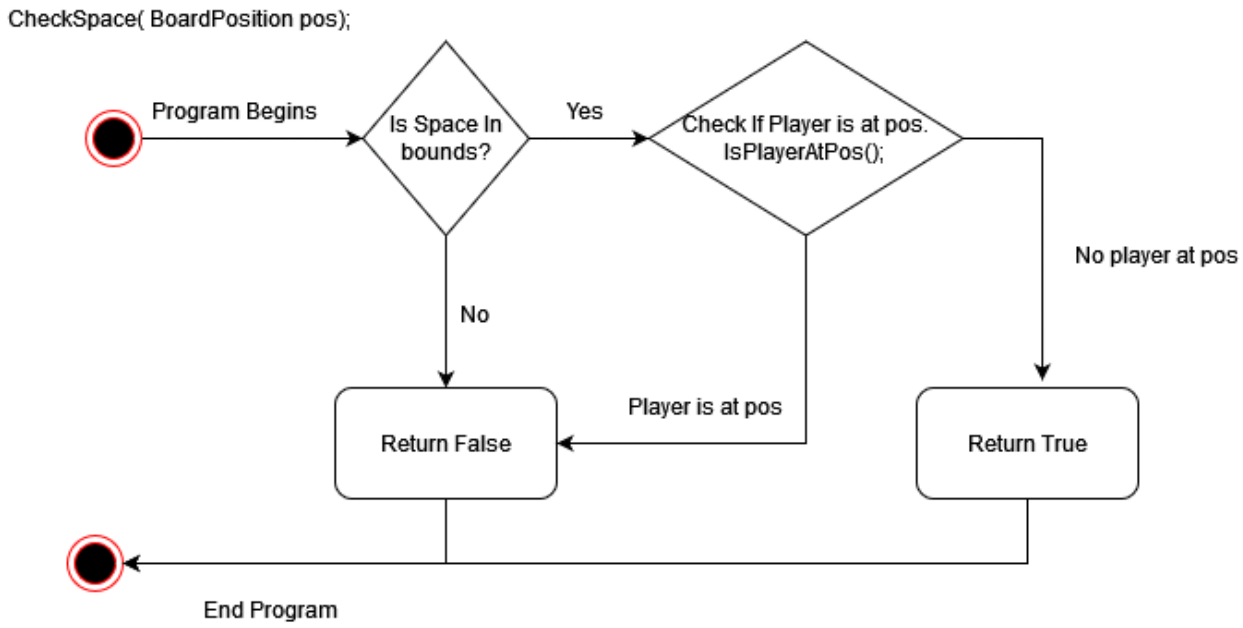
Activity Diagrams

GameScreen Activity Diagram:

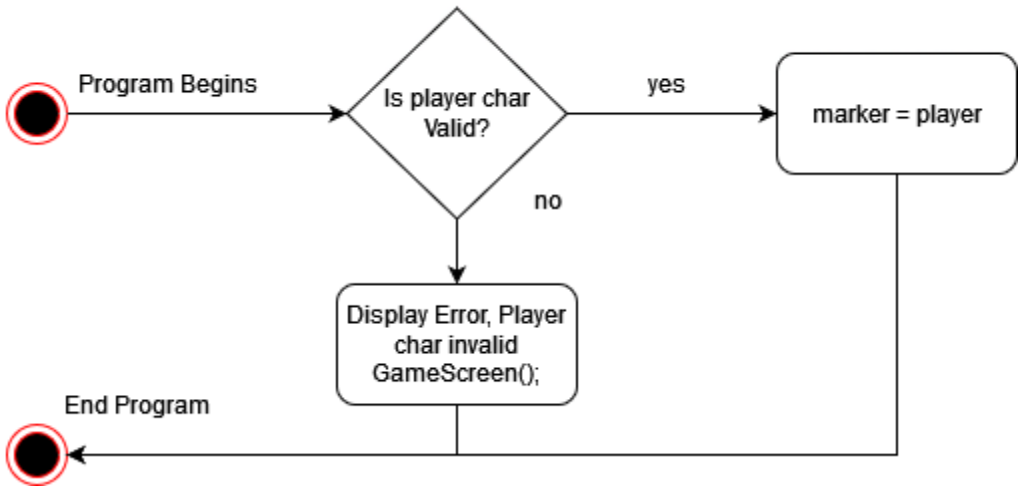
Main:



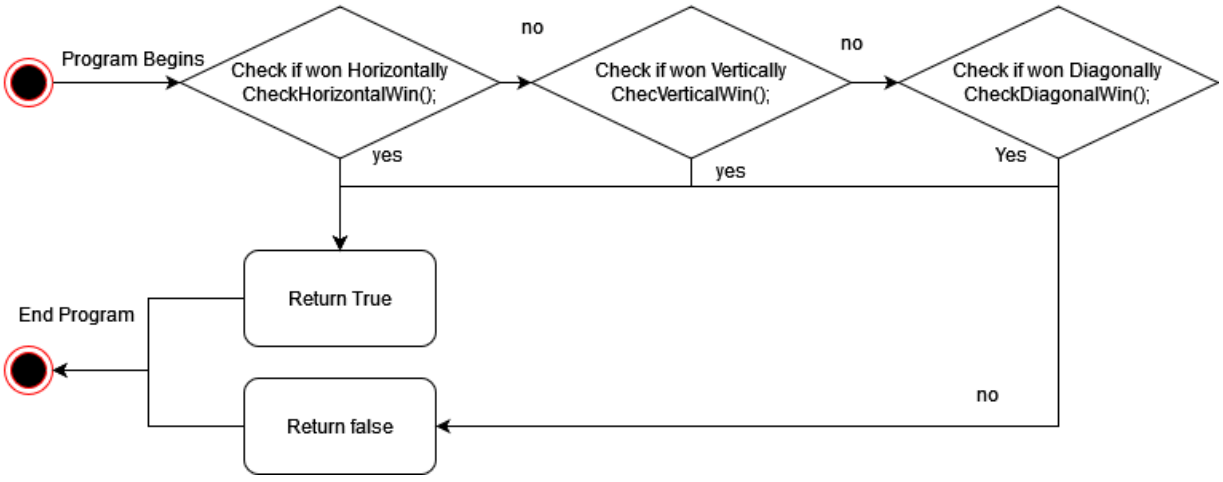
GameBoard Activity Diagrams:



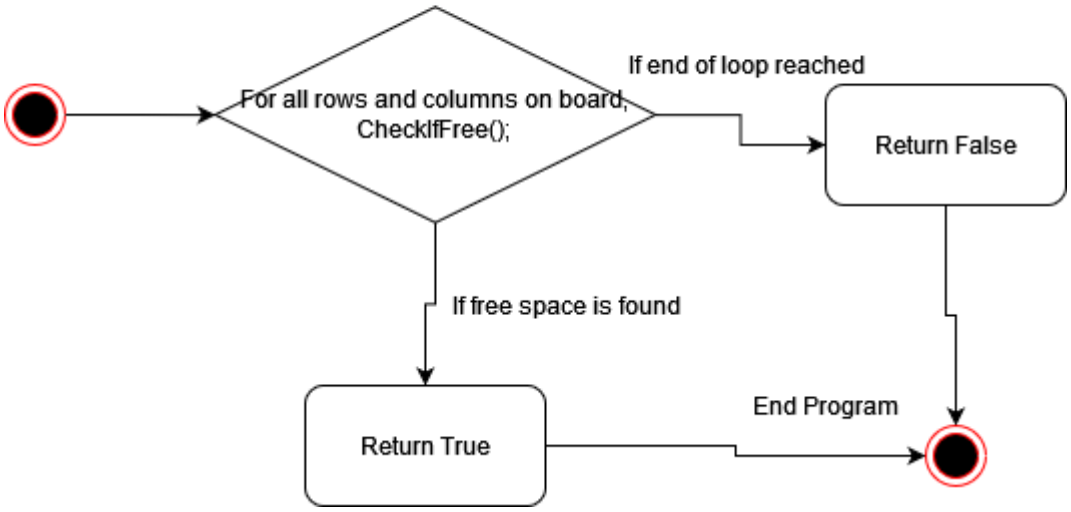
```
void placeMarker(BoardPosition marker, char player)
```



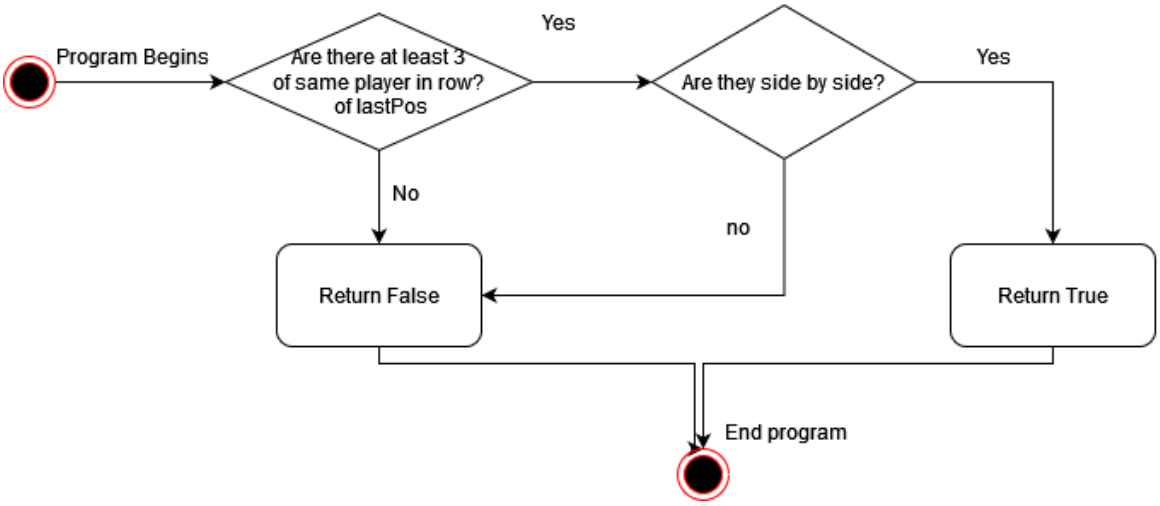

```
public boolean checkForWinner(BoardPosition lastPos)
```



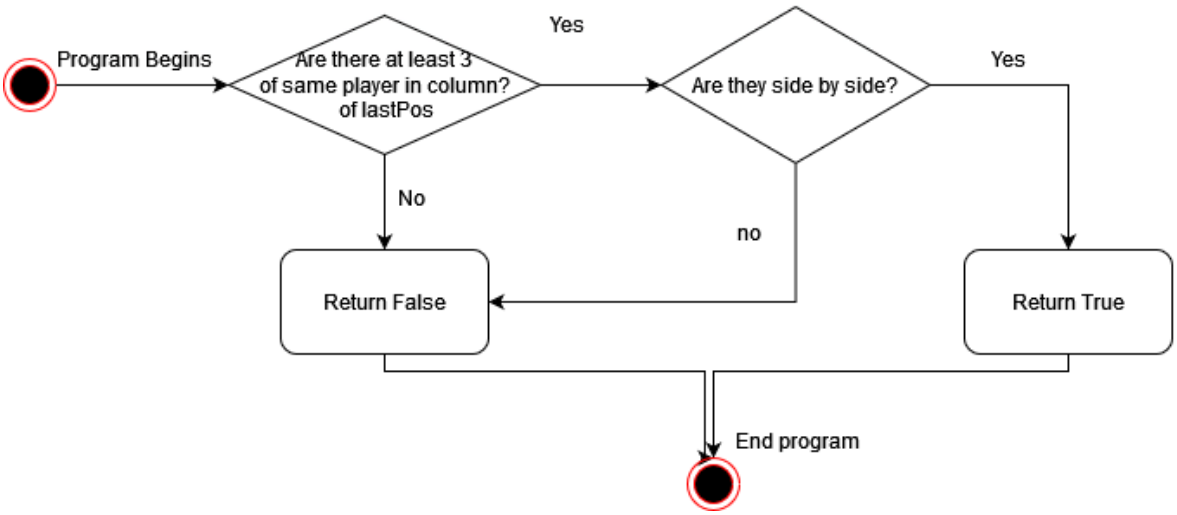
```
public boolean checkForDraw()
```



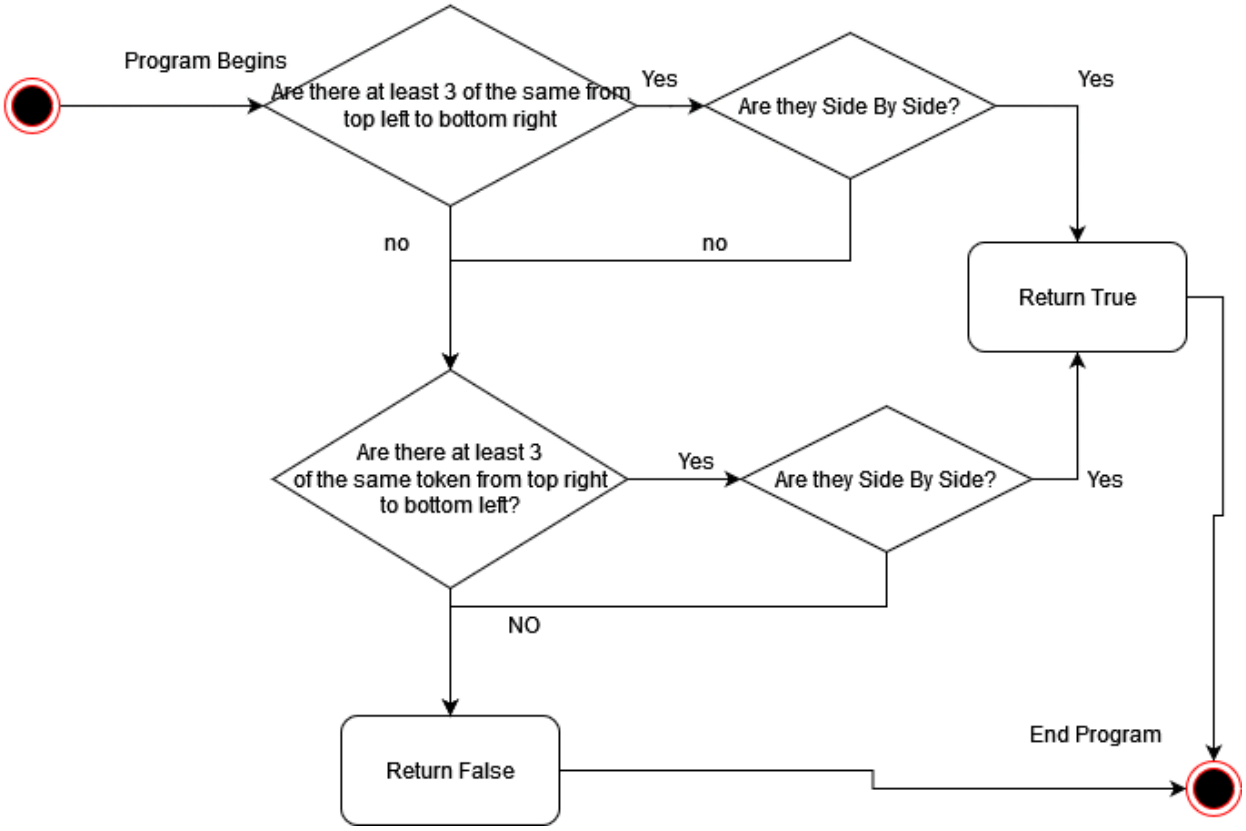

```
publicboolean checkHorizontalWin(BoardPosition lastPos, char player)
```



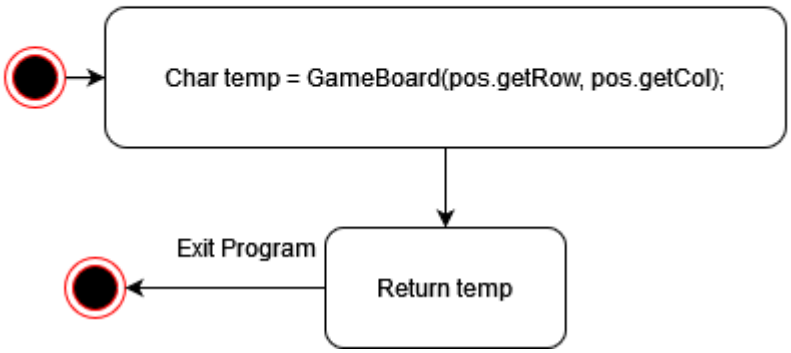
```
publicboolean checkVerticalWin(BoardPosition lastPos, char player)
```



```
publicboolean checkDiagonalWin(BoardPosition lastPos, char player)
```



```
public char whatsAtPos(BoardPosition pos)
```




```
boolean isPlayerAtPos(BoardPosition pos, char player)
```