Requirement Analysis:

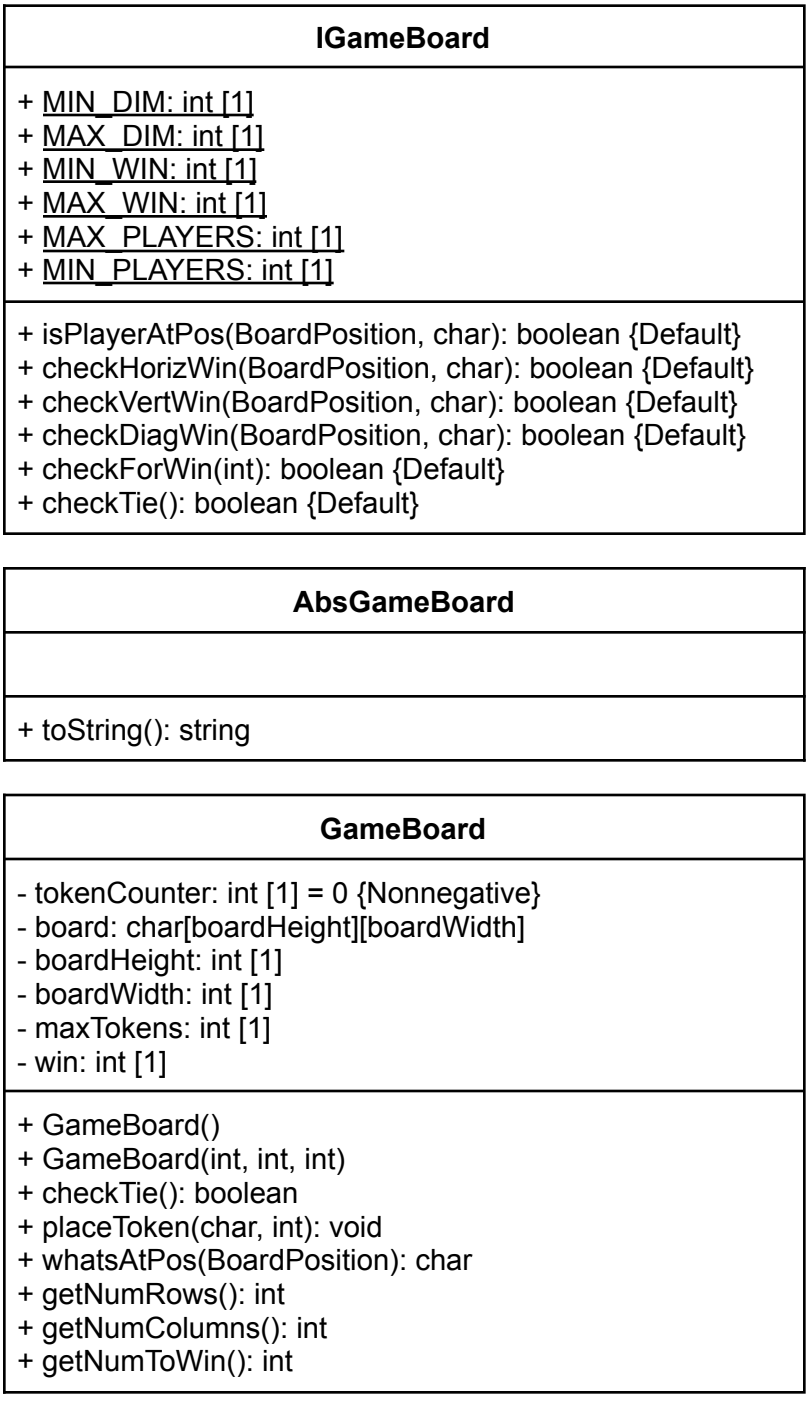
Functional Requirements:

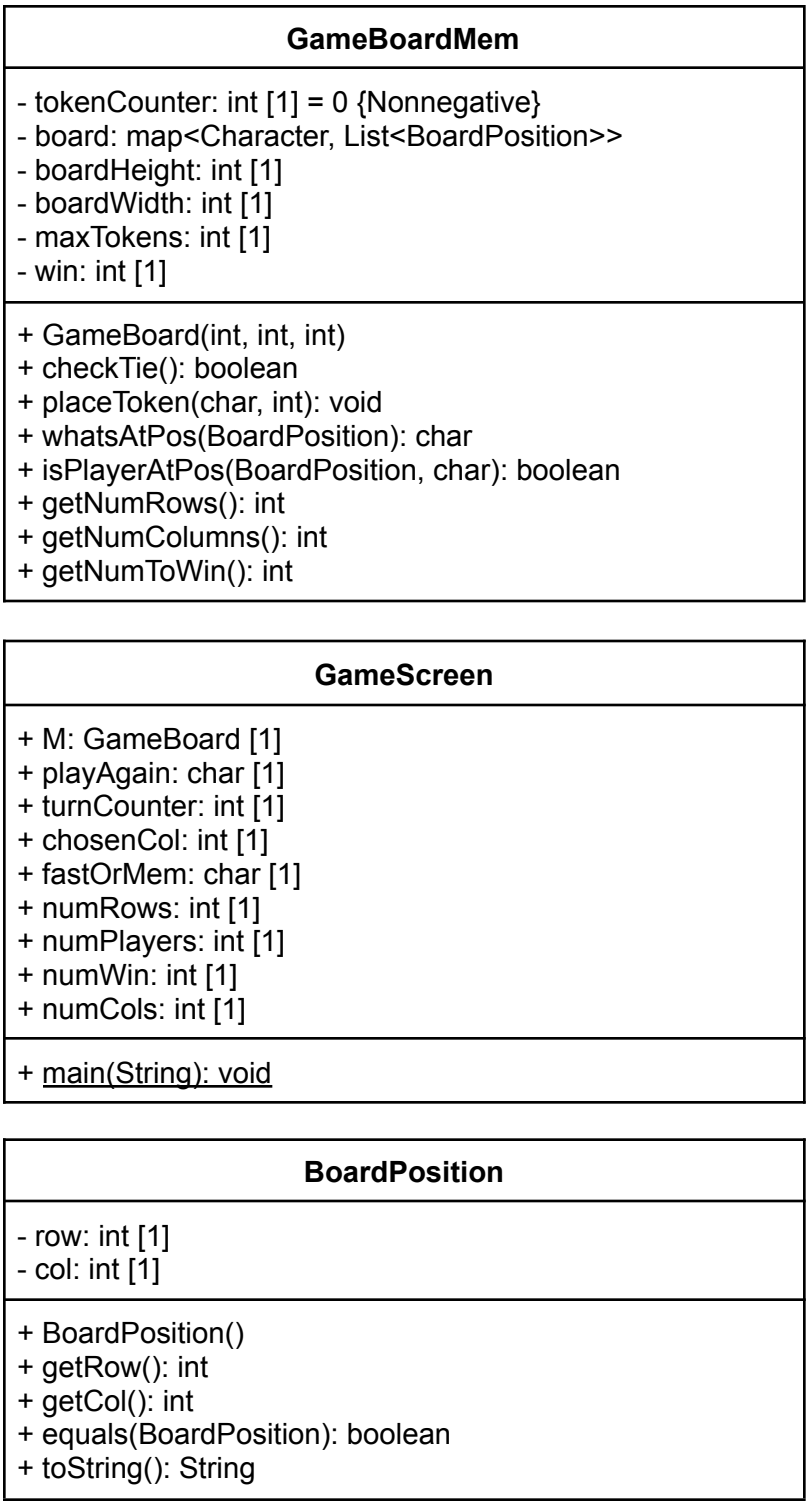
- As a player, I can choose a player that can be any character other than ‘ ‘ to determine which tokens I will use.
- As a player, I can place a token in a chosen column to try and connect 5 horizontally.
- As a player, I can place a token in a chosen column to try and connect 5 vertically.
- As a player, I can place a token in a chosen column to try and connect 5 diagonally.
- As a player, I can place enough tokens to reach the max count which will result in a tie condition.
- As a player, I can decide whether or not to play again at the end of a game by typing ‘y’ or ‘n’.
- As a player, I swap turns with another player, placing a token after the other player does so that the game is fair.
- As a player, I can select any column to place a token in, but if I am out of bounds when placing I will be prompted to place again, so that I do not waste a turn.
- As a player, I can choose the number of rows that the game board will have to add another dimension of variability to the game.
- As a player, I can choose the number of columns that the game board will have to add another dimension of variability to the game.
- As a player, I can choose the number of tokens needed to win the game to add another dimension of variability to the game.
- As a player, I can choose a faster version of the game or a memory efficient version of the game to play.
- As a player, I have access to seeing the game board to determine my next token placement.
- The game must accept column integer input from the user.
- The game must check to make sure that a column is not full when a player tries to place a token.
- The game must check to see if a player has won by connecting 5 tokens in a row horizontally, vertically, or diagonally.

Non-Functional Requirements:

- The game executable must run on Unix.
- The game must be between 3x3 and 100x100 board of characters.
- The game must allow player 1 to go first.
- The game must allow <0>,<0> to be the bottom left board position on the board.
- The game code must be able to compile in Java 11.
- The game code must be able to run in Java 11.

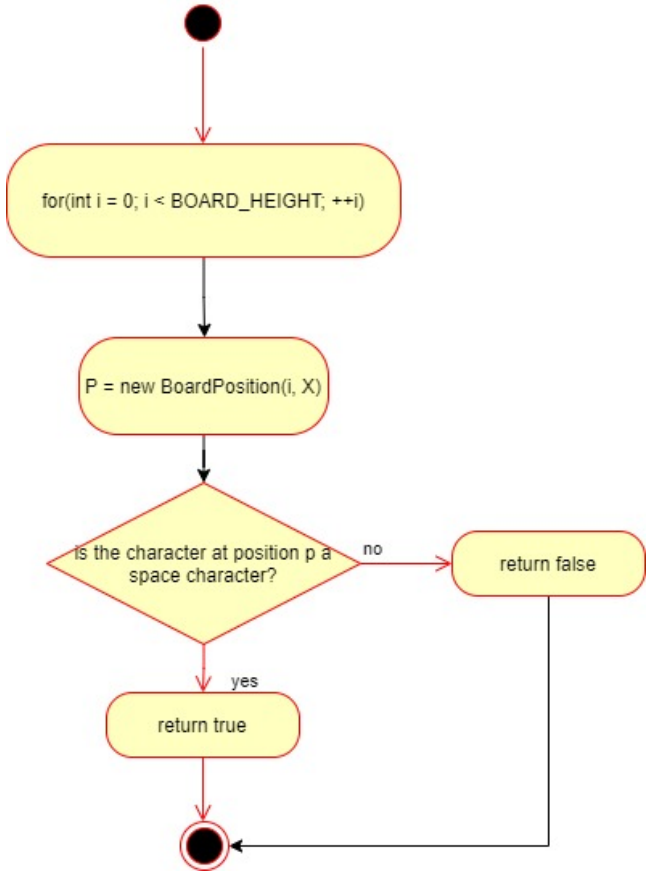
UML Class Diagrams:



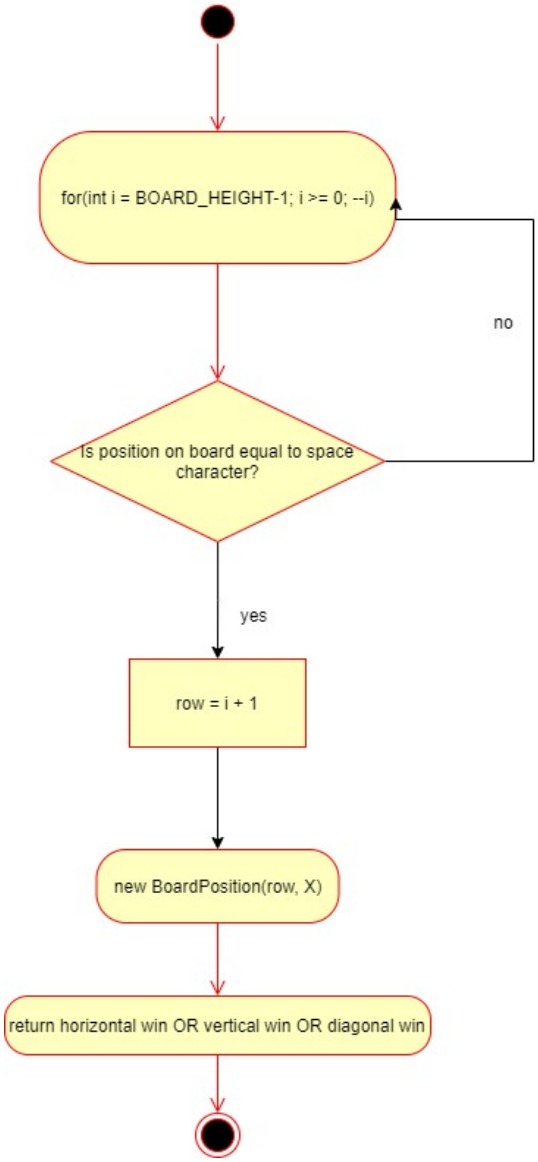

GameBoardMem
- tokenCounter: int [1] = 0 {Nonnegative}
- board: map<Character, List<BoardPosition>>
- boardHeight: int [1]
- boardWidth: int [1]
- maxTokens: int [1]
- win: int [1]
+ GameBoard(int, int, int)
+ checkTie(): boolean
+ placeToken(char, int): void
+ whatsAtPos(BoardPosition): char
+ isPlayerAtPos(BoardPosition, char): boolean
+ getNumRows(): int
+ getNumColumns(): int
+ getNumToWin(): int
GameScreen
+ M: GameBoard [1]
+ playAgain: char [1]
+ turnCounter: int [1]
+ chosenCol: int [1]
+ fastOrMem: char [1]
+ numRows: int [1]
+ numPlayers: int [1]
+ numWin: int [1]
+ numCols: int [1]
+ main(String): void
BoardPosition
- row: int [1]
- col: int [1]
+ BoardPosition()
+ getRow(): int
+ getCol(): int
+ equals(BoardPosition): boolean
+ toString(): String

UML Activity Diagrams:

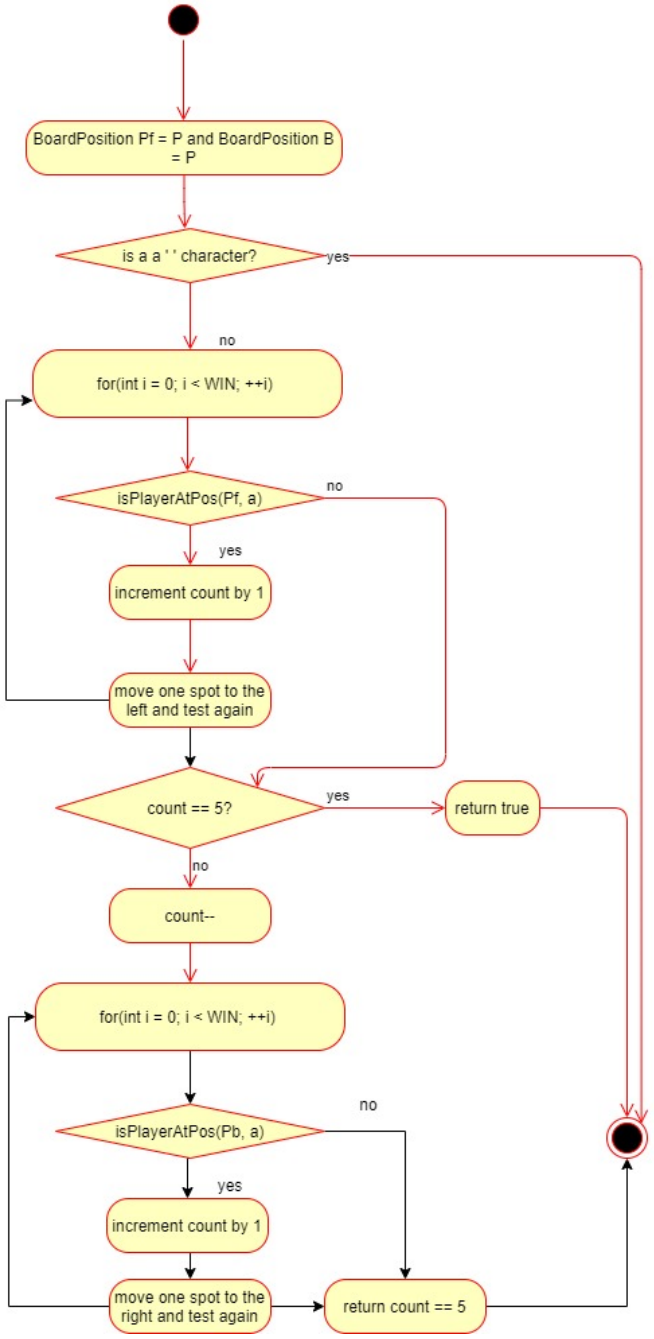
checkIfFree(int): boolean {default}



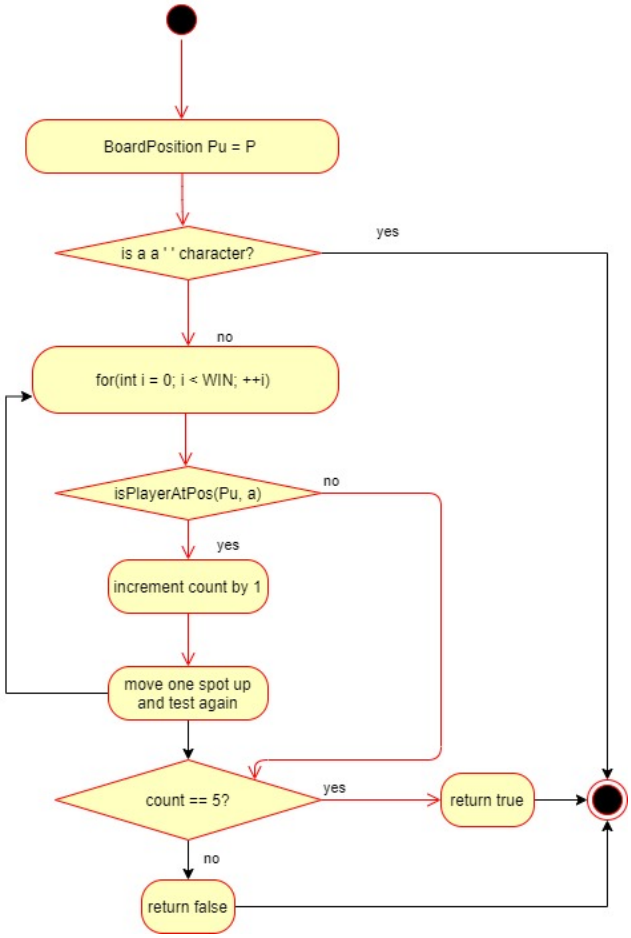
checkForWin(int): boolean {default}



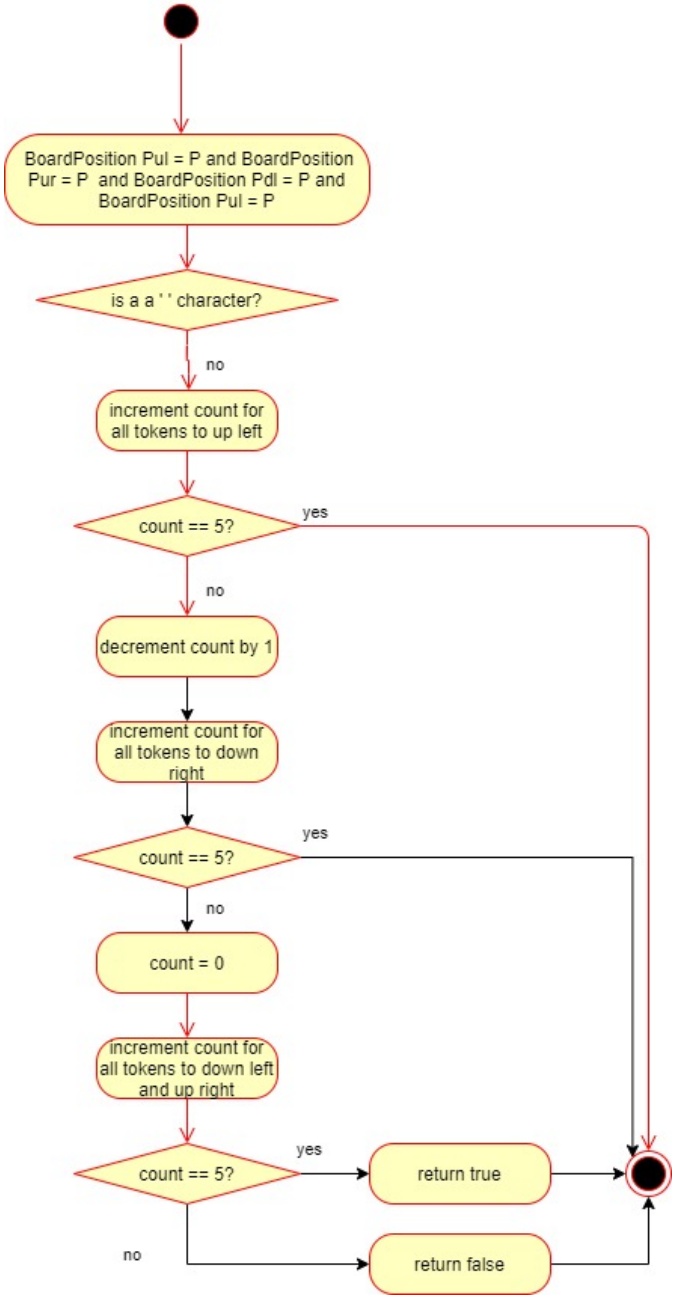
checkHorizWin(BoardPosition, char): boolean {default}



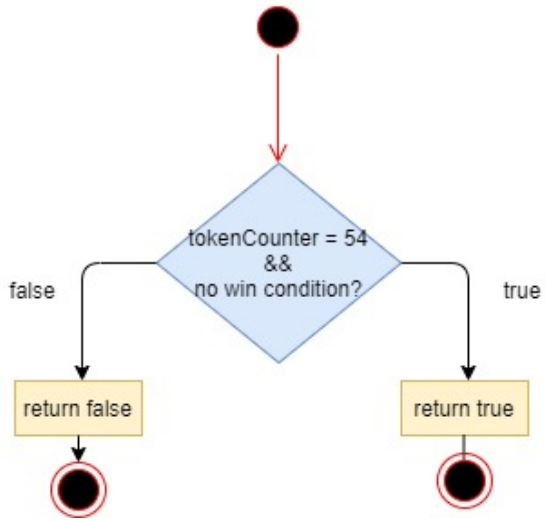
checkVertWin(BoardPosition, char): boolean {default}



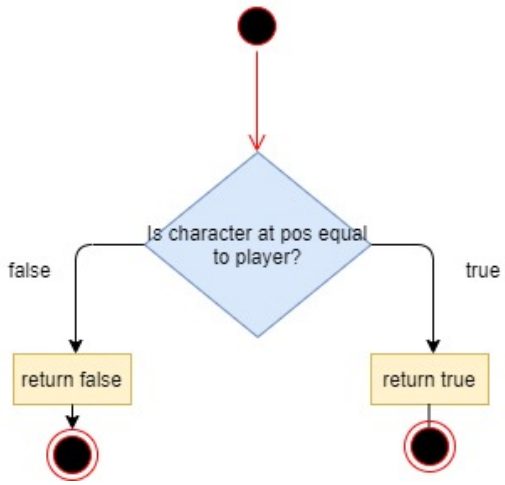
checkDiagWin(BoardPosition, char): boolean {default}



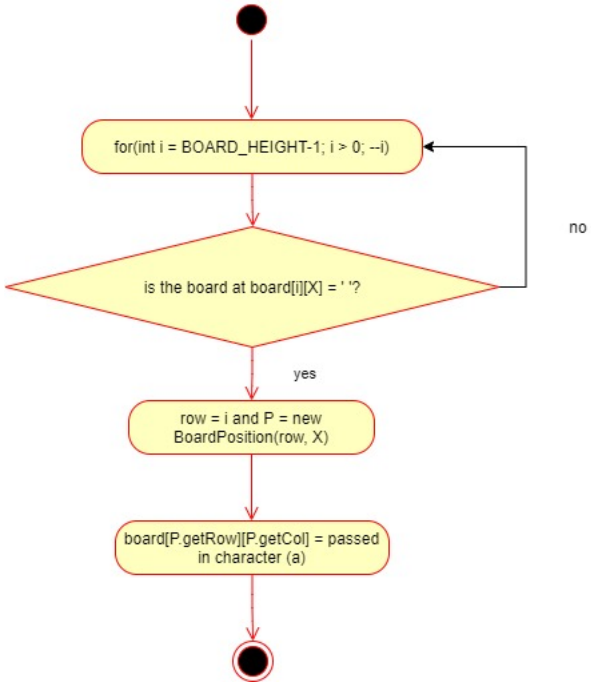
checkTie(): boolean (GameBoard and GameBoardMem)



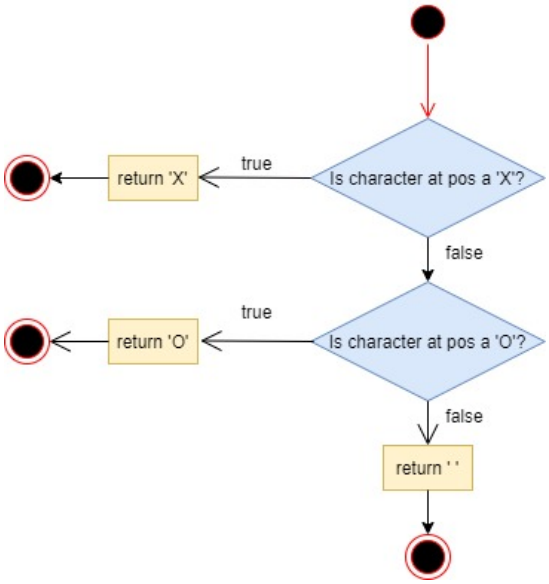
isPlayerAtPos(BoardPosition, char): boolean {default}



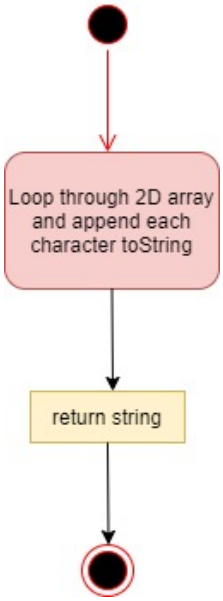
placeToken(char, int): void (GameBoard)



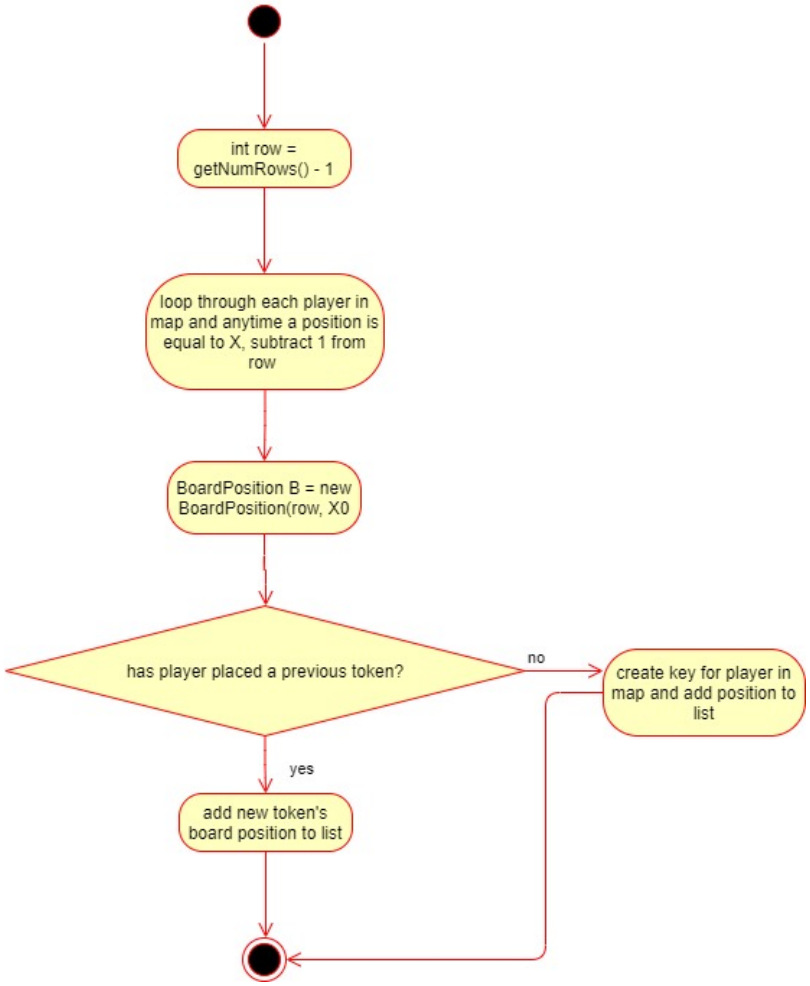
whatsAtPos(BoardPosition): char (GameBoard)



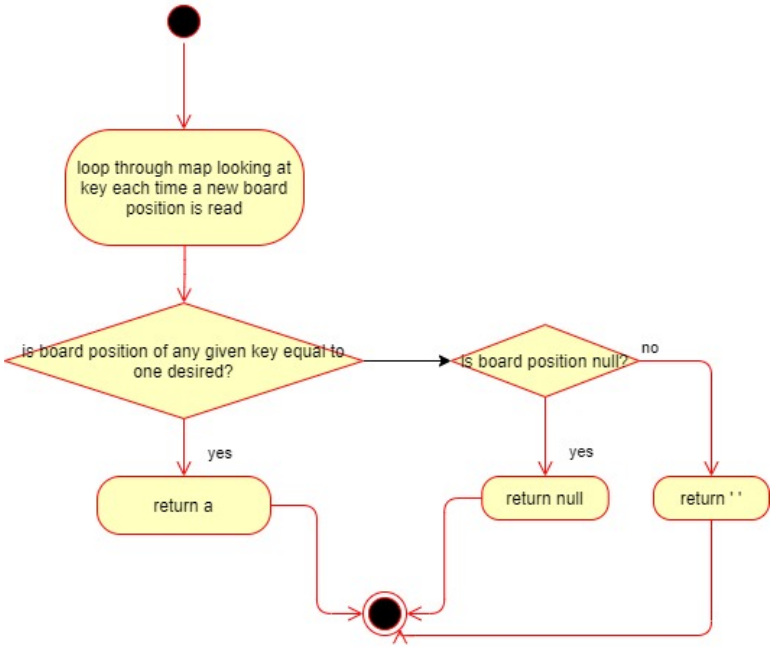
toString(): String



placeToken(Character, int): void (GameBoardMem)



whatsAtPos(BoardPosition): Character (GameBoardMem)



isPlayerAtPos(BoardPosition, char): boolean (GameBoardMem)