Sean Danielson
10/17/19

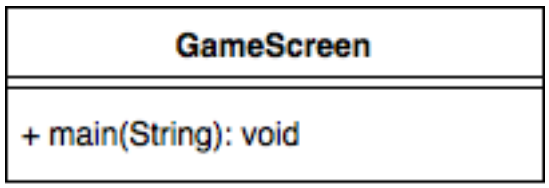
TicTacToe Project Report

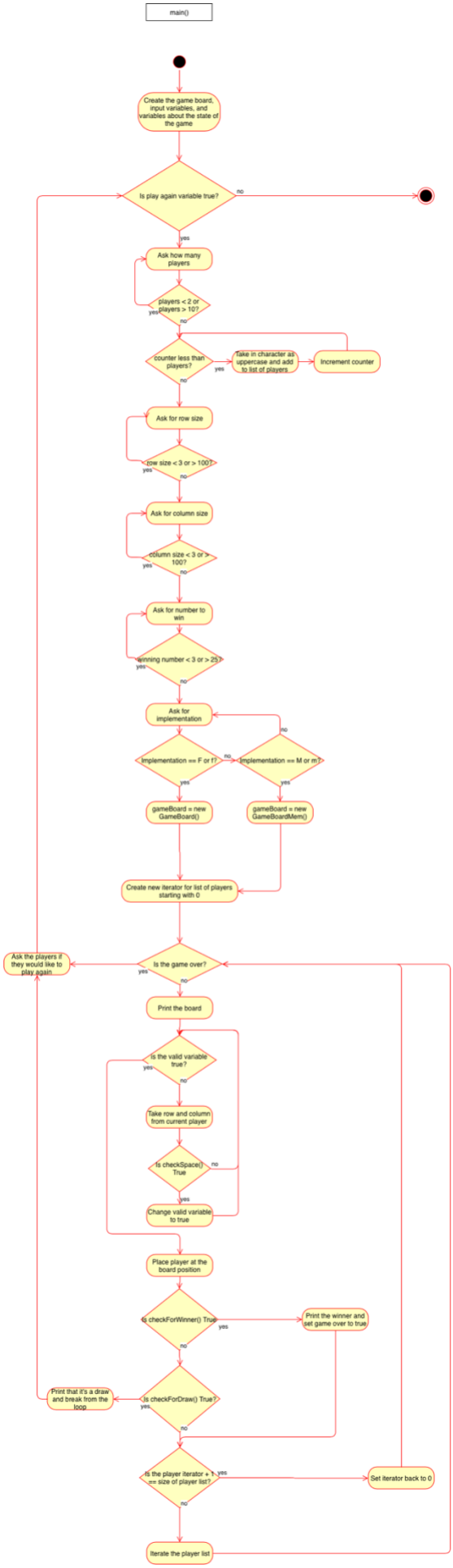
Requirements Analysis

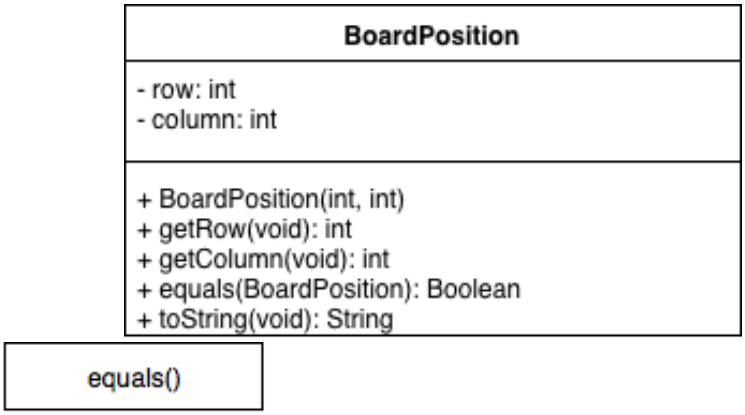
- **Functional Requirements**
  - **As a player, I can easily tell what player's turn it is so I can play on my turn.**
  - **As a player, I can see the game board, so I know what is available and what is not.**
  - **As a player, I can specify whether or not I want to play again so that I can play multiple games.**
  - **As a player, I understand how the coordinate system works so I can input my moves correctly.**
  - **As a player, I can pick again if I pick an unavailable space, so I don't lose my turn**
  - **As a player, if I get the defined winning number in a row horizontally, I will win the game, so I can win the game**
  - **As a player, if I get the defined winning number in a row vertically, I will win the game, so I can win the game**
  - **As a player, if I get the defined winning number in a row diagonally, I will win the game, so I can win the game**
  - **As a player, I can end the game in a tie by taking the last space on the board without getting the winning number in a row, so the game can end**
  - **As a player, I can input the bounds of the board, so that each game is different.**
  - **As a player, I can input the number required in a row to win.**
  - **As a player, I can input the number of players in the game so that I can play with multiple players.**
  - **As a player, I can decide what character to represent myself as on the gameboard so I can be different from other players.**
  - **As a player, I can decide whether or not I want a faster implementation or a memory efficient implementation so I can make tradeoffs for my games.**
  - **As a player, I can reenter the rules of the game if I decide to play again, so I can play multiple games with different settings.**
- **Nonfunctional Requirements**
  - **The system checks for winning moves frequently.**
  - **The system checks for wins in all directions.**
  - **The system checks for draws frequently.**
  - **The system does not have a user interface.**
  - **The system is direct in its prompts.**
  - **The system uses the command line for user interaction.**
  - **The system checks for player character conflicts.**
  - **The system is developed in Java.**
  - **The system was created with Intellij IDEA.**
  - **The system runs on Unix.**
  - **The system codes to the interface of IGameBoard for ease of game board creation between memory efficient and speed efficient implementations.**
  - **The system keeps track of whose turn it is.**
  - **0,0 is the top left of the board**
  - **The board size cannot exceed 100x100.**
  - **The board size cannot be lower than 3x3**

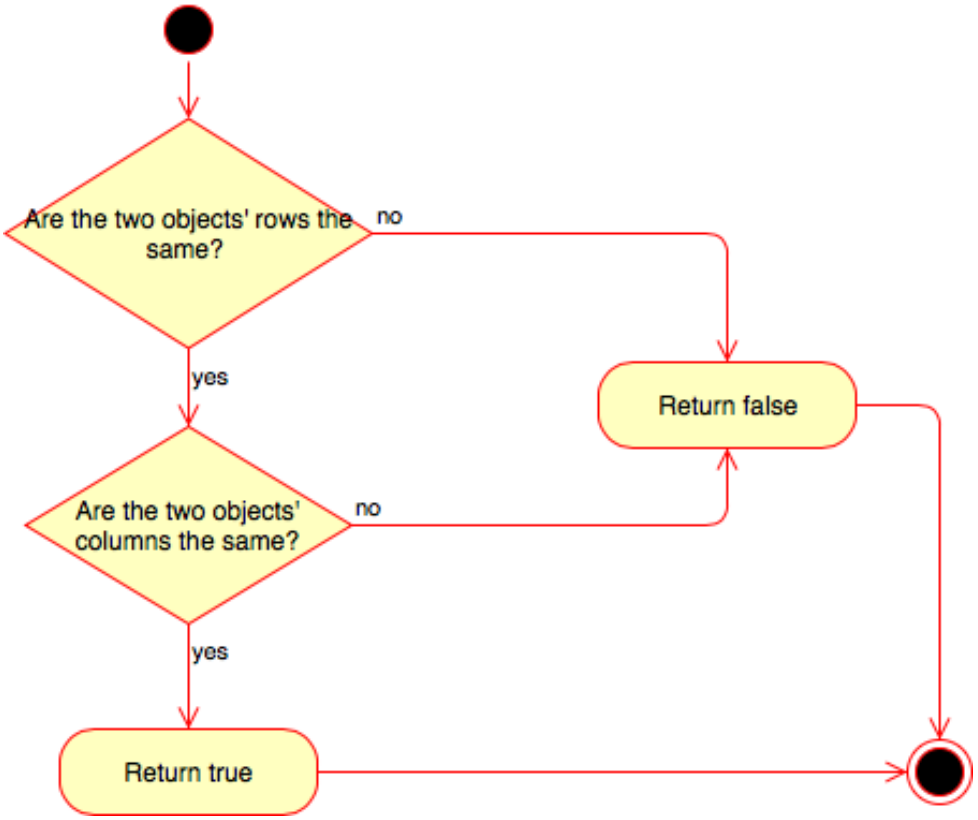
- **The number to win cannot exceed 25**
- **The number to win cannot be lower than 3.**
- **The number of players must be between 2 and 10 inclusive.**

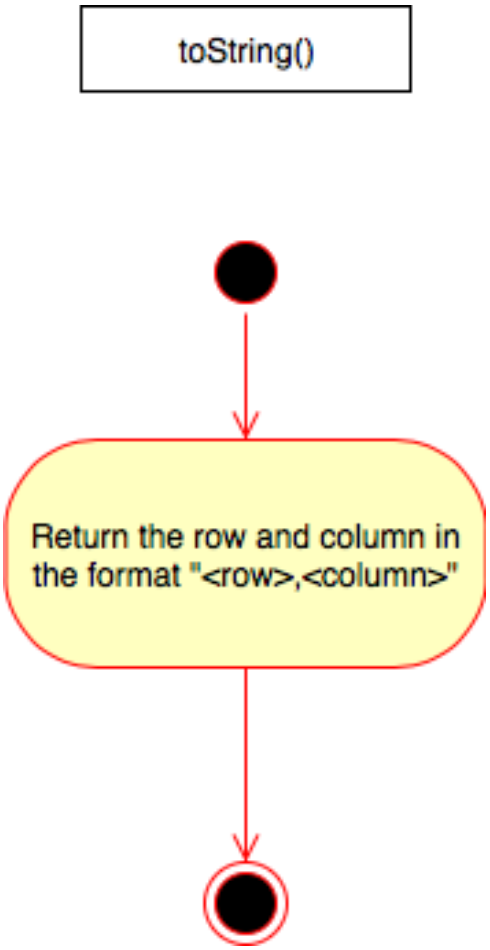
Design

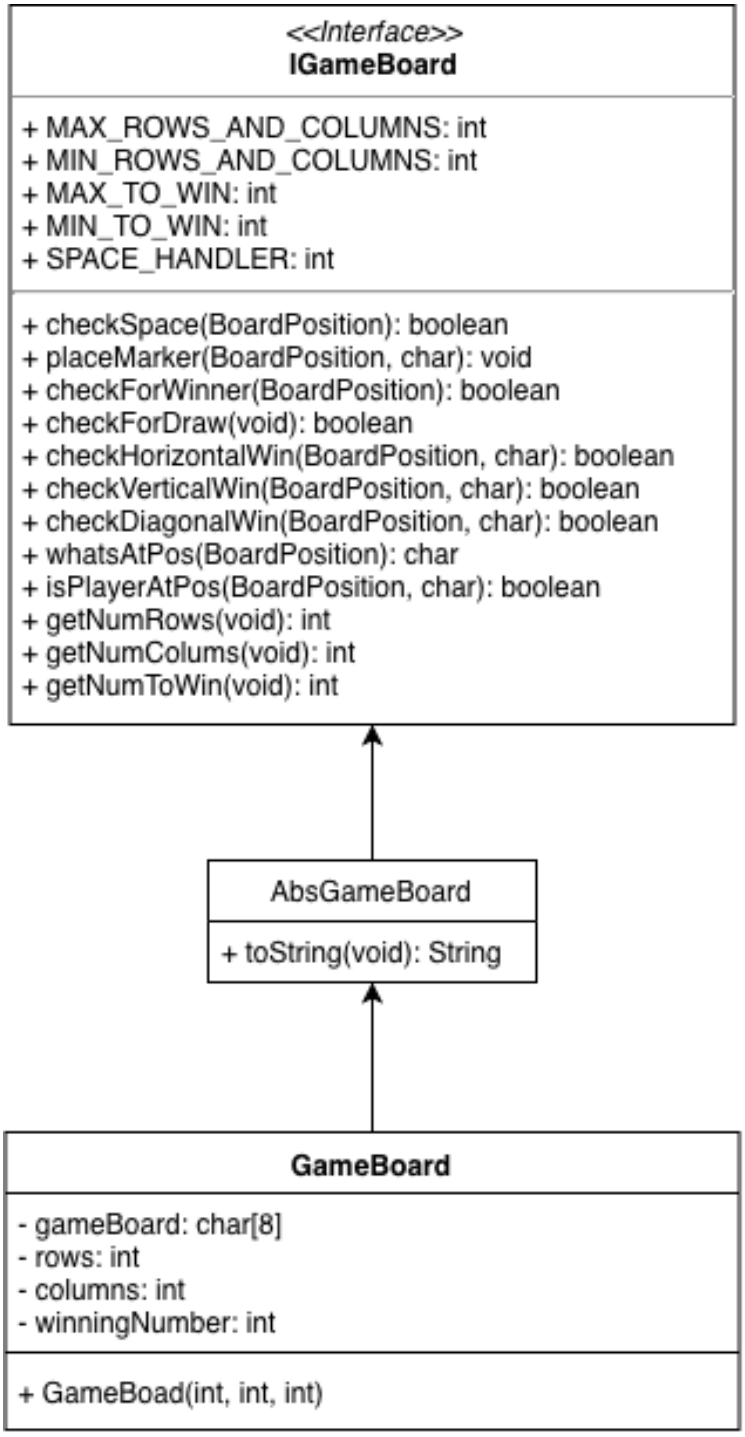


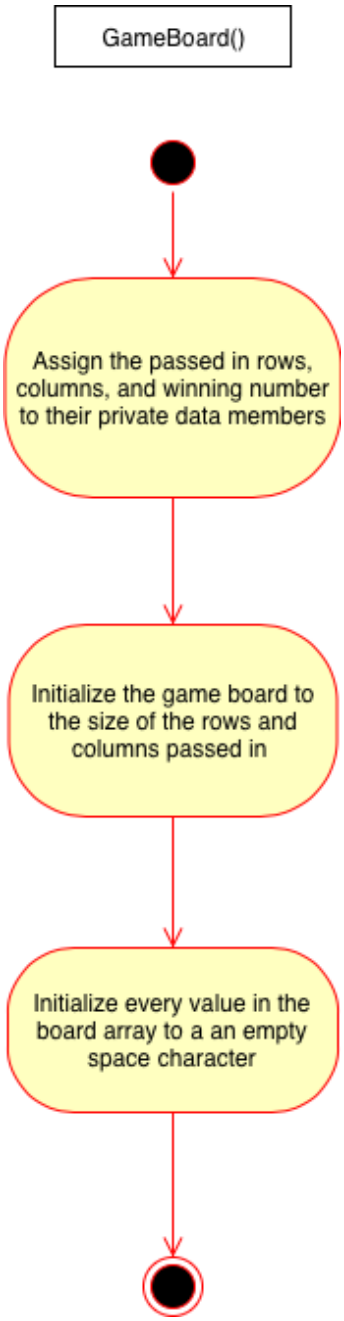

main()
Create the game board, input variables, and variables about the state of the game
Is play again variable true?
no
yes
Ask how many players
players < 2 or players > 10?
yes
no
counter less than players?
yes
Take in character as uppercase and add to list of players
Increment counter
no
Ask for row size
row size < 3 or > 100?
yes
no
Ask for column size
column size < 3 or > 100?
yes
no
Ask for number to win
winning number < 3 or > 25?
yes
no
Ask for implementation
Implementation == F or f?
no
Implementation == M or m?
no
yes
yes
gameBoard = new GameBoard()
gameBoard = new GameBoardMem()
Create new iterator for list of players starting with 0
Is the game over?
yes
Ask the players if they would like to play again
no
Print the board
Is the valid variable true?
yes
no
Take row and column from current player
Is checkSpace() True
no
yes
Change valid variable to true
Place player at the board position
Is checkForWinner() True
yes
Print the winner and set game over to true
no
Is checkForDraw() True?
yes
Print that it's a draw and break from the loop
no
Is the player iterator + 1 == size of player list?
yes
Set iterator back to 0
no
Iterate the player list

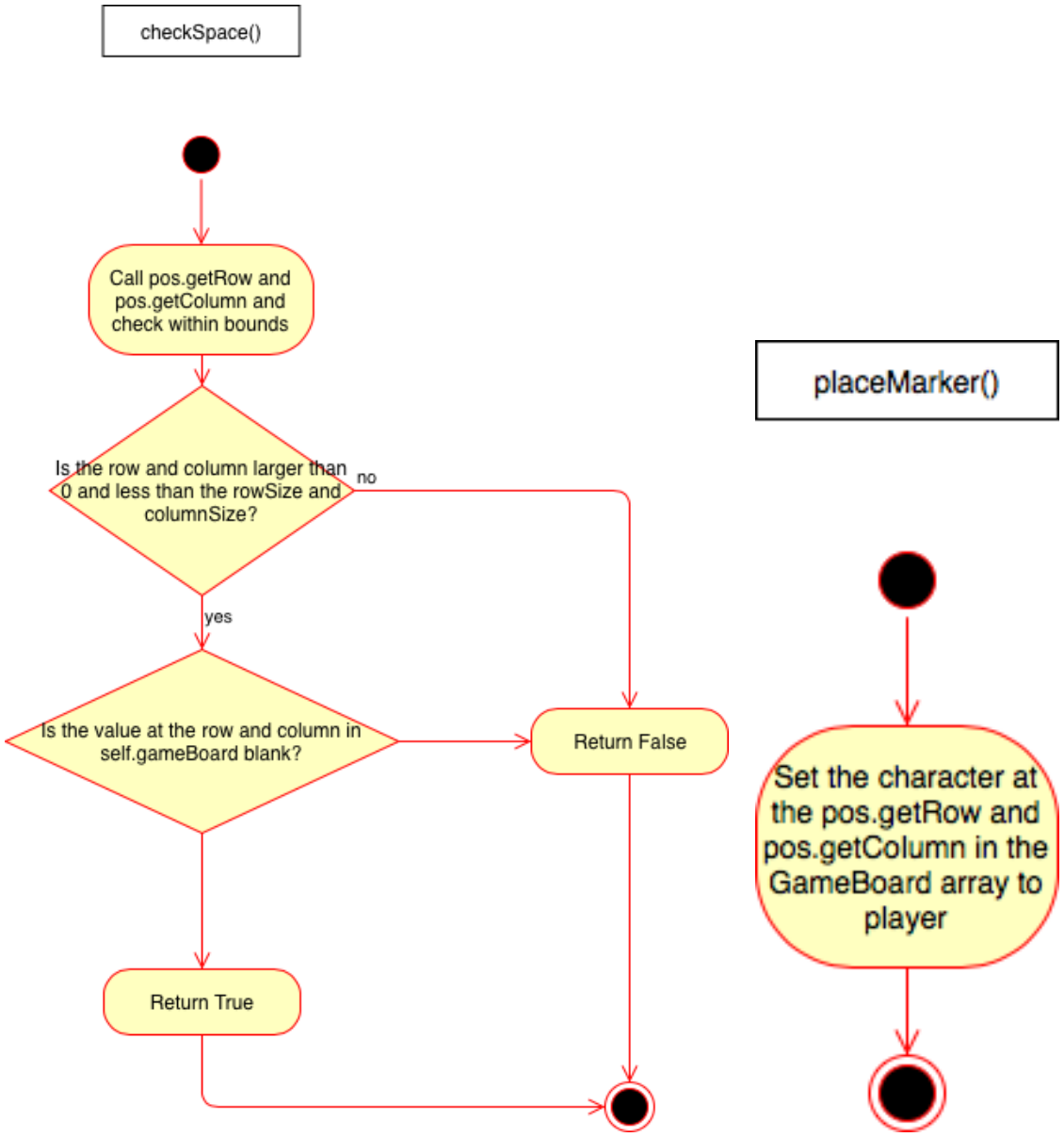

BoardPosition
- row: int
- column: int
+ BoardPosition(int, int)
+ getRow(void): int
+ getColumn(void): int
+ equals(BoardPosition): Boolean
+ toString(void): String
equals()

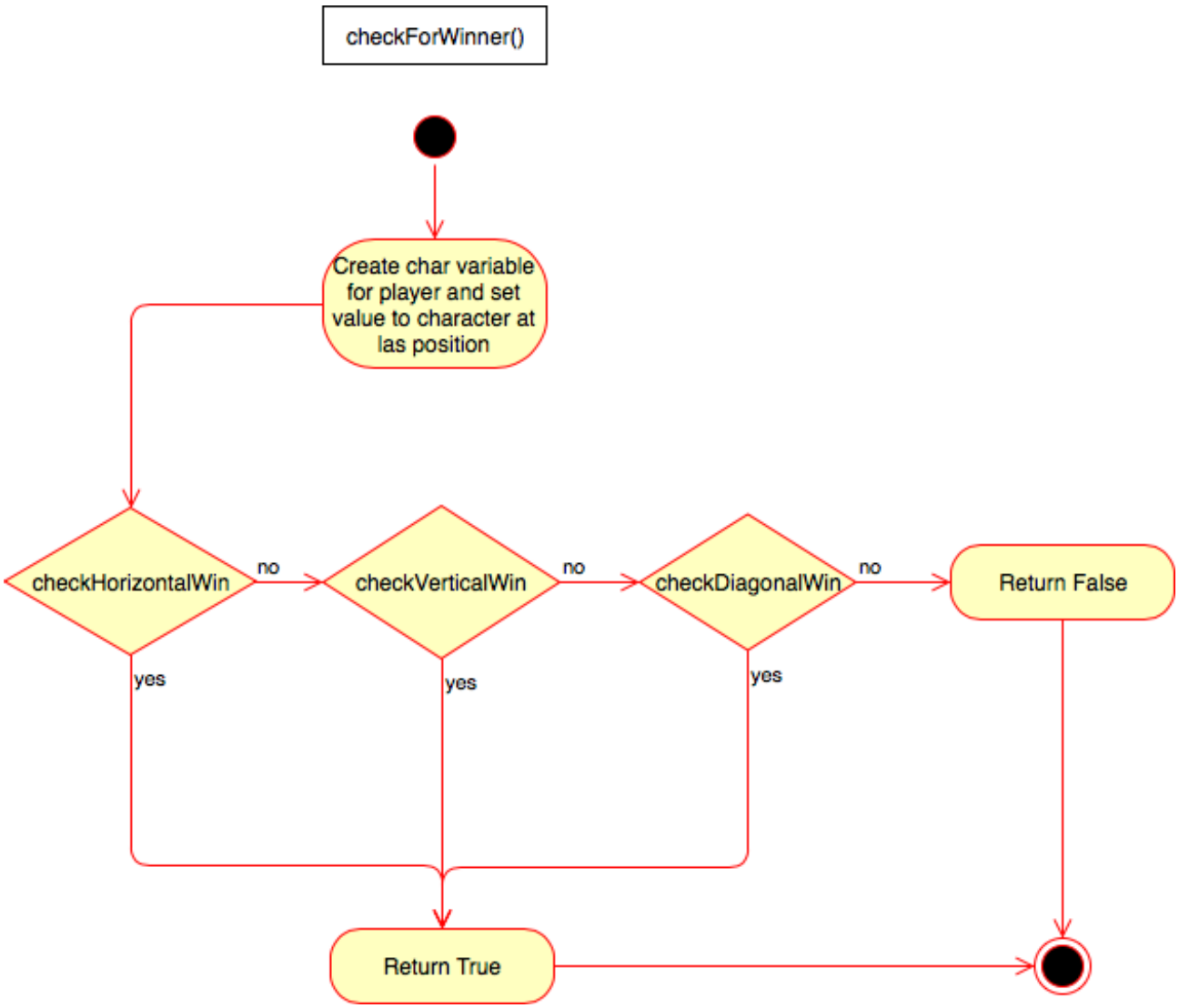

Are the two objects' rows the same?
no
yes
Return false
Are the two objects' columns the same?
no
yes
Return true

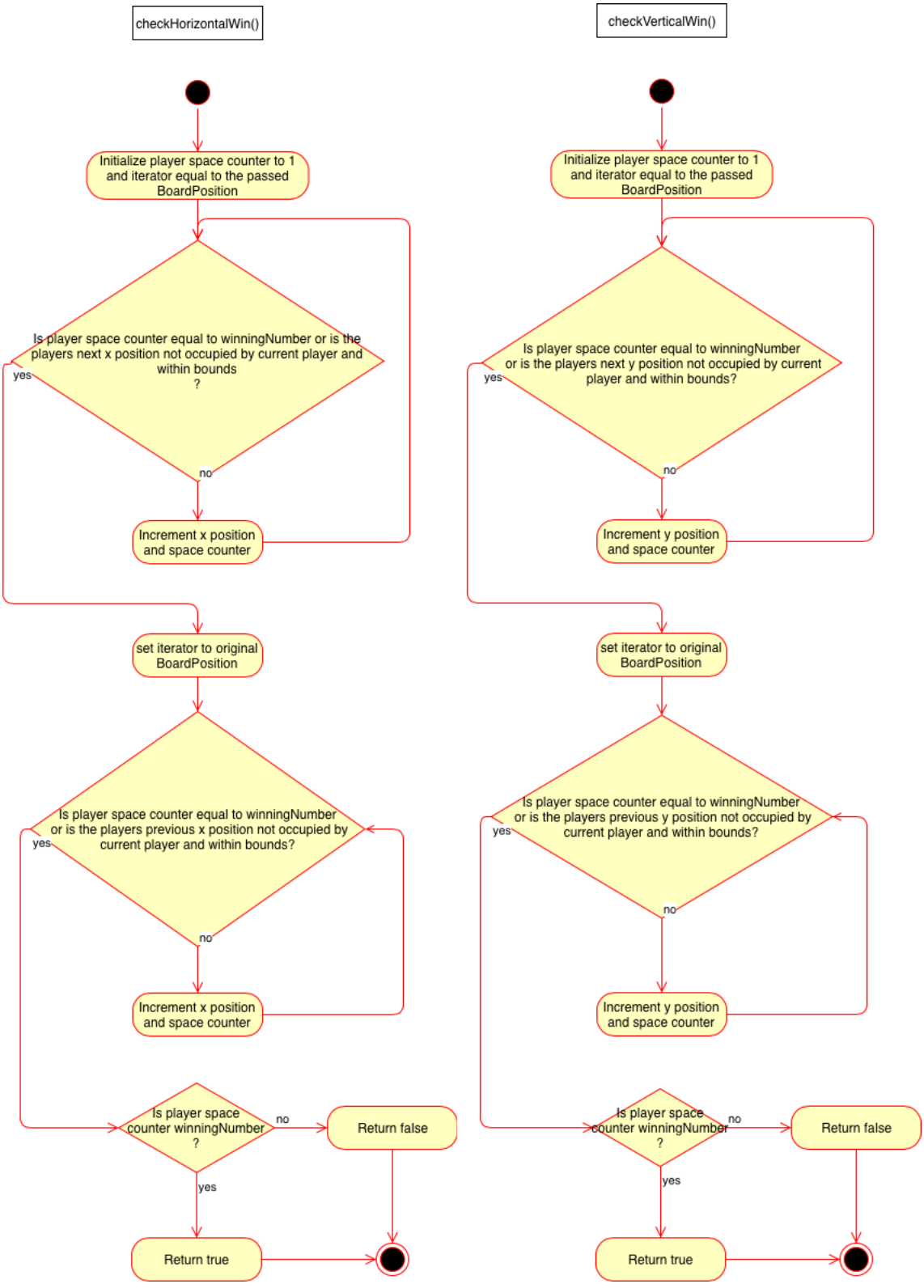

toString()
Return the row and column in the format "<row>,<column>"

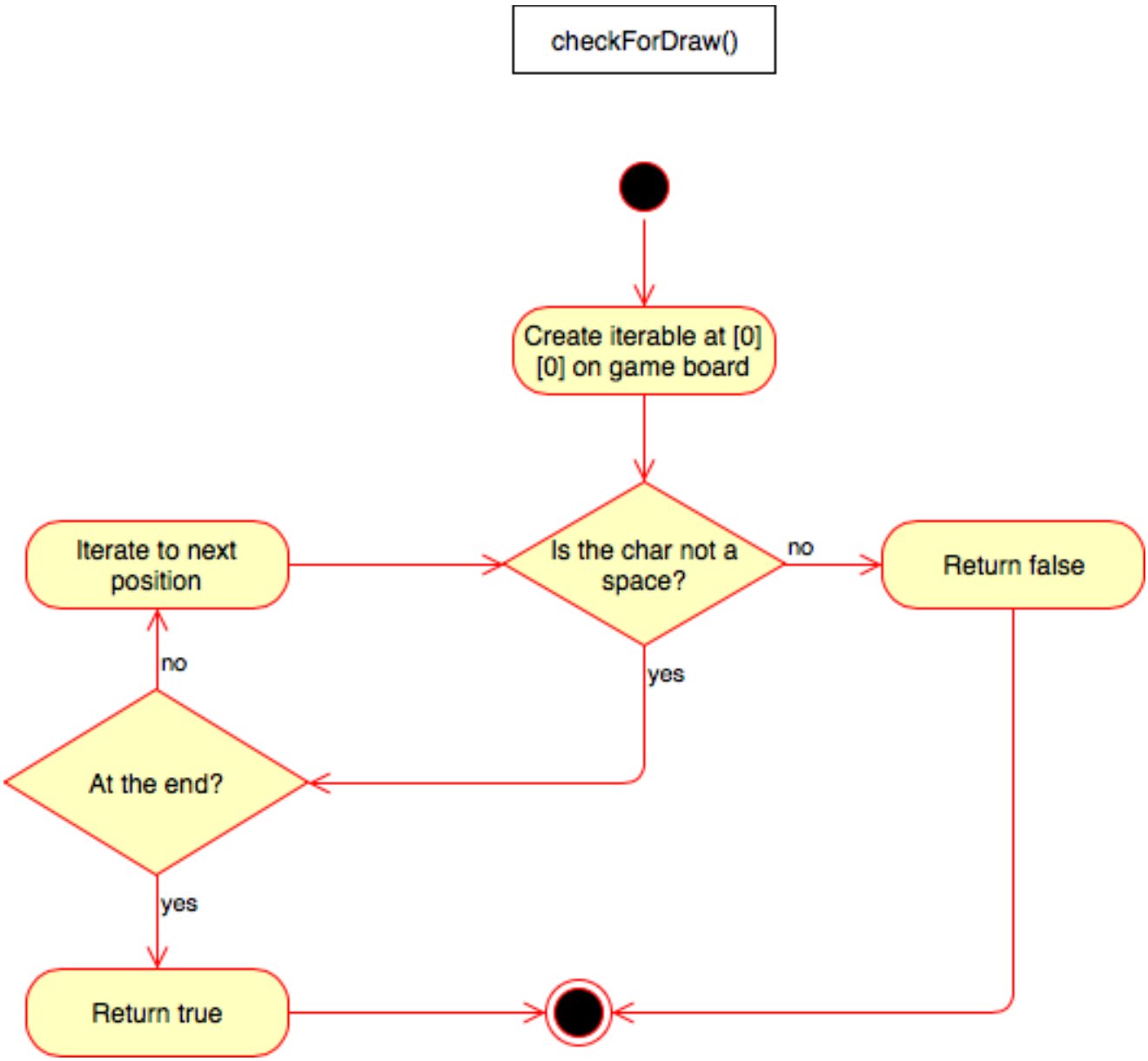

<<Interface>>
IGameBoard
+ MAX_ROWS_AND_COLUMNS: int
+ MIN_ROWS_AND_COLUMNS: int
+ MAX_TO_WIN: int
+ MIN_TO_WIN: int
+ SPACE_HANDLER: int
+ checkSpace(BoardPosition): boolean
+ placeMarker(BoardPosition, char): void
+ checkForWinner(BoardPosition): boolean
+ checkForDraw(void): boolean
+ checkHorizontalWin(BoardPosition, char): boolean
+ checkVerticalWin(BoardPosition, char): boolean
+ checkDiagonalWin(BoardPosition, char): boolean
+ whatsAtPos(BoardPosition): char
+ isPlayerAtPos(BoardPosition, char): boolean
+ getNumRows(void): int
+ getNumColums(void): int
+ getNumToWin(void): int
AbsGameBoard
+ toString(void): String
GameBoard
- gameBoard: char[8]
- rows: int
- columns: int
- winningNumber: int
+ GameBoad(int, int, int)

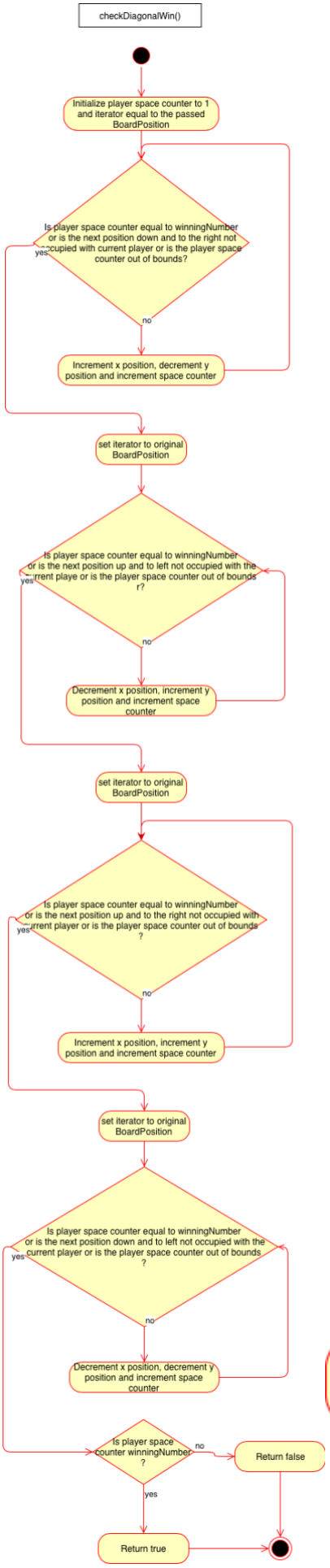

GameBoard()
Assign the passed in rows, columns, and winning number to their private data members
Initialize the game board to the size of the rows and columns passed in
Initialize every value in the board array to a an empty space character

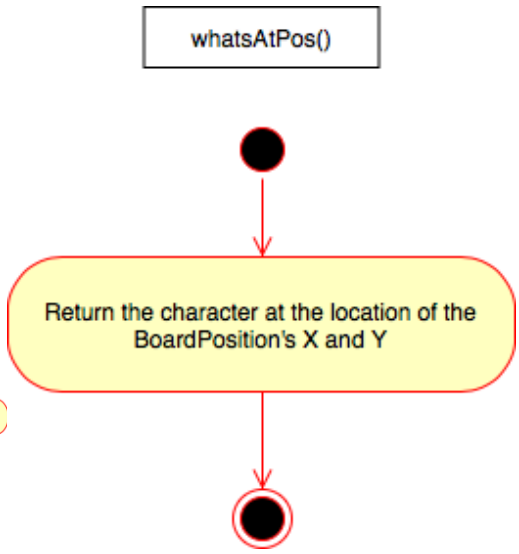

checkSpace()
Call pos.getRow and pos.getColumn and check within bounds
Is the row and column larger than 0 and less than the rowSize and columnSize?
no
yes
Is the value at the row and column in self.gameBoard blank?
Return False
Return True
placeMarker()
Set the character at the pos.getRow and pos.getColumn in the GameBoard array to player

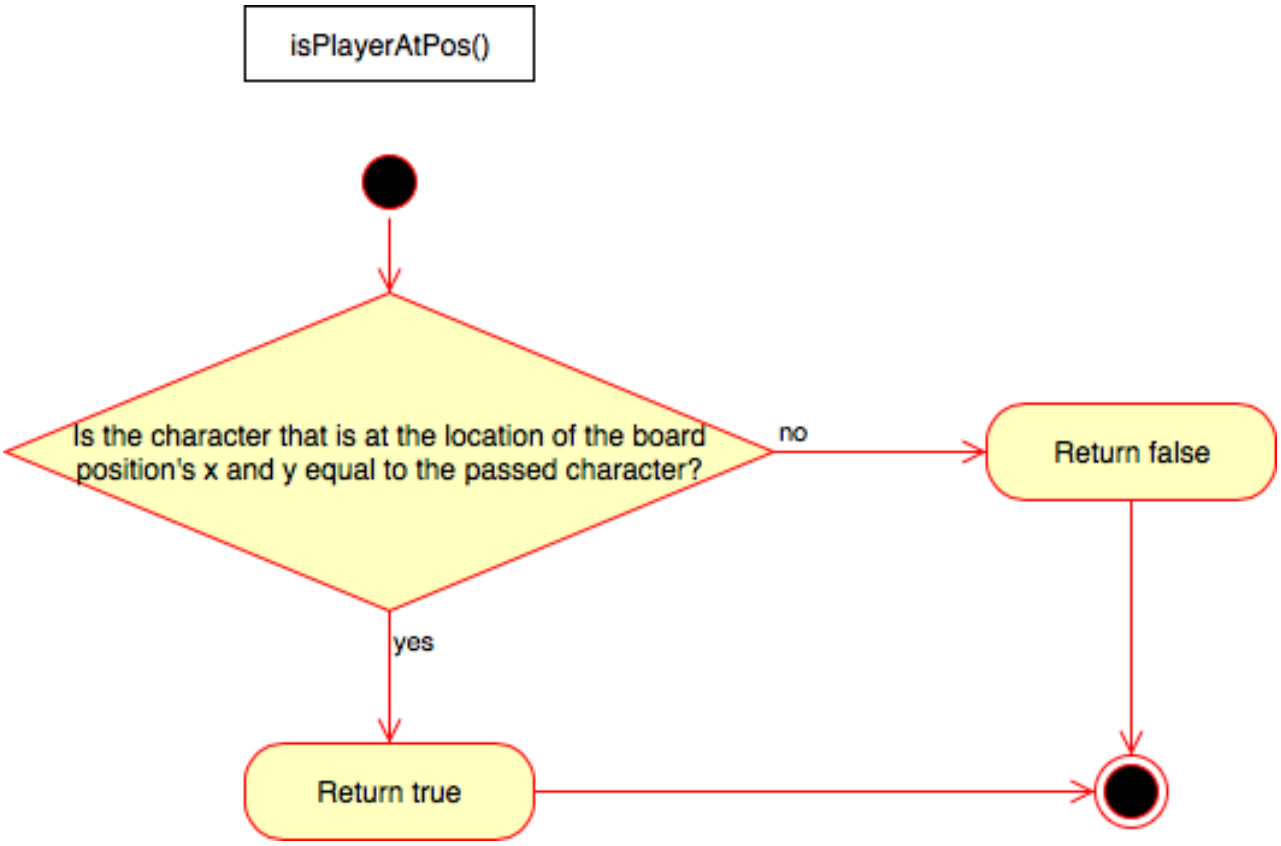

checkForWinner()
Create char variable for player and set value to character at las position
checkHorizontalWin
no
checkVerticalWin
no
checkDiagonalWin
no
Return False
yes
yes
yes
Return True

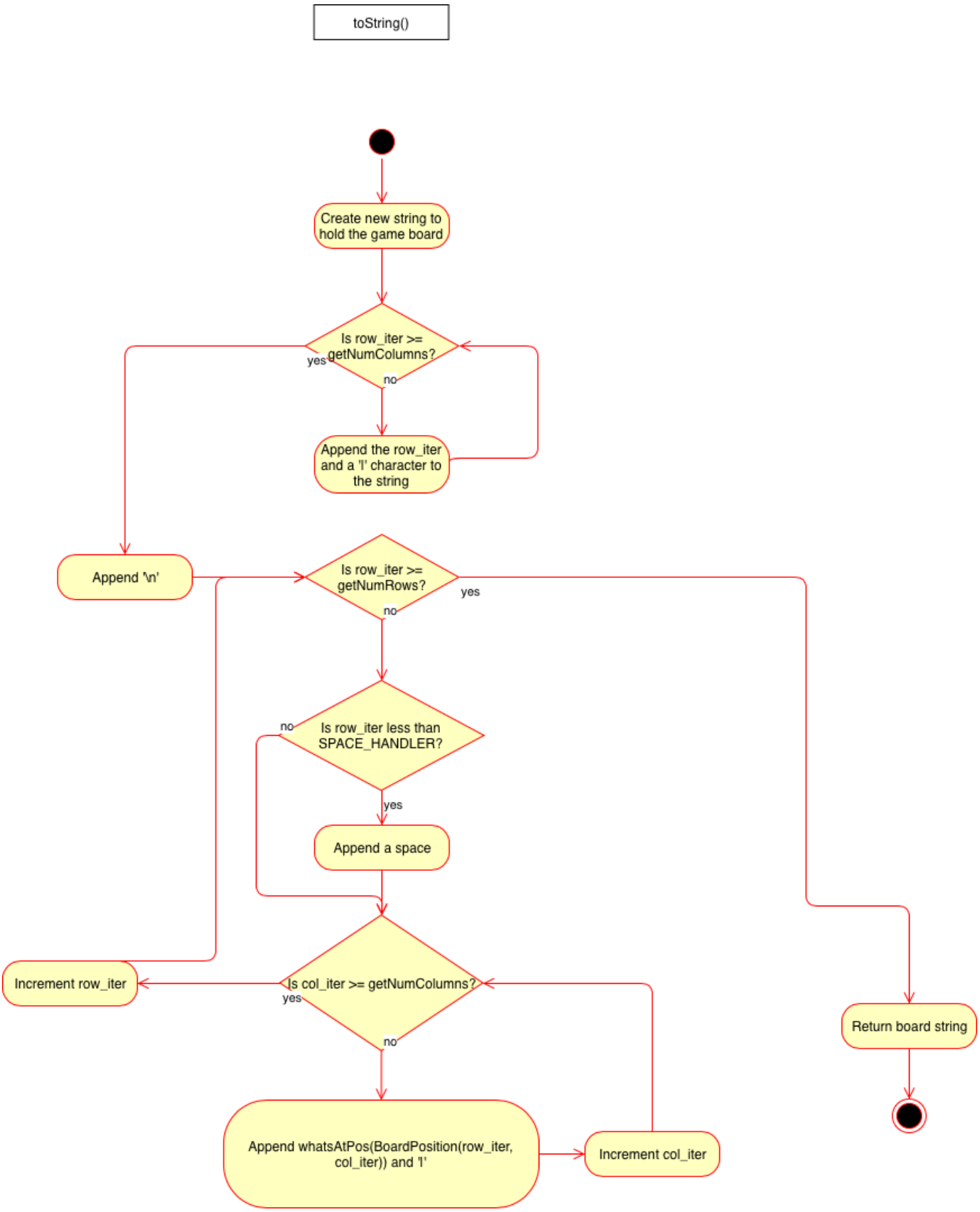

checkHorizontalWin()
Initialize player space counter to 1 and iterator equal to the passed BoardPosition
Is player space counter equal to winningNumber or is the players next x position not occupied by current player and within bounds ?
yes
no
Increment x position and space counter
set iterator to original BoardPosition
Is player space counter equal to winningNumber or is the players previous x position not occupied by current player and within bounds?
yes
no
Increment x position and space counter
Is player space counter winningNumber ?
no
Return false
yes
Return true
checkVerticalWin()
Initialize player space counter to 1 and iterator equal to the passed BoardPosition
Is player space counter equal to winningNumber or is the players next y position not occupied by current player and within bounds?
yes
no
Increment y position and space counter
set iterator to original BoardPosition
Is player space counter equal to winningNumber or is the players previous y position not occupied by current player and within bounds?
yes
no
Increment y position and space counter
Is player space counter winningNumber ?
no
Return false
yes
Return true

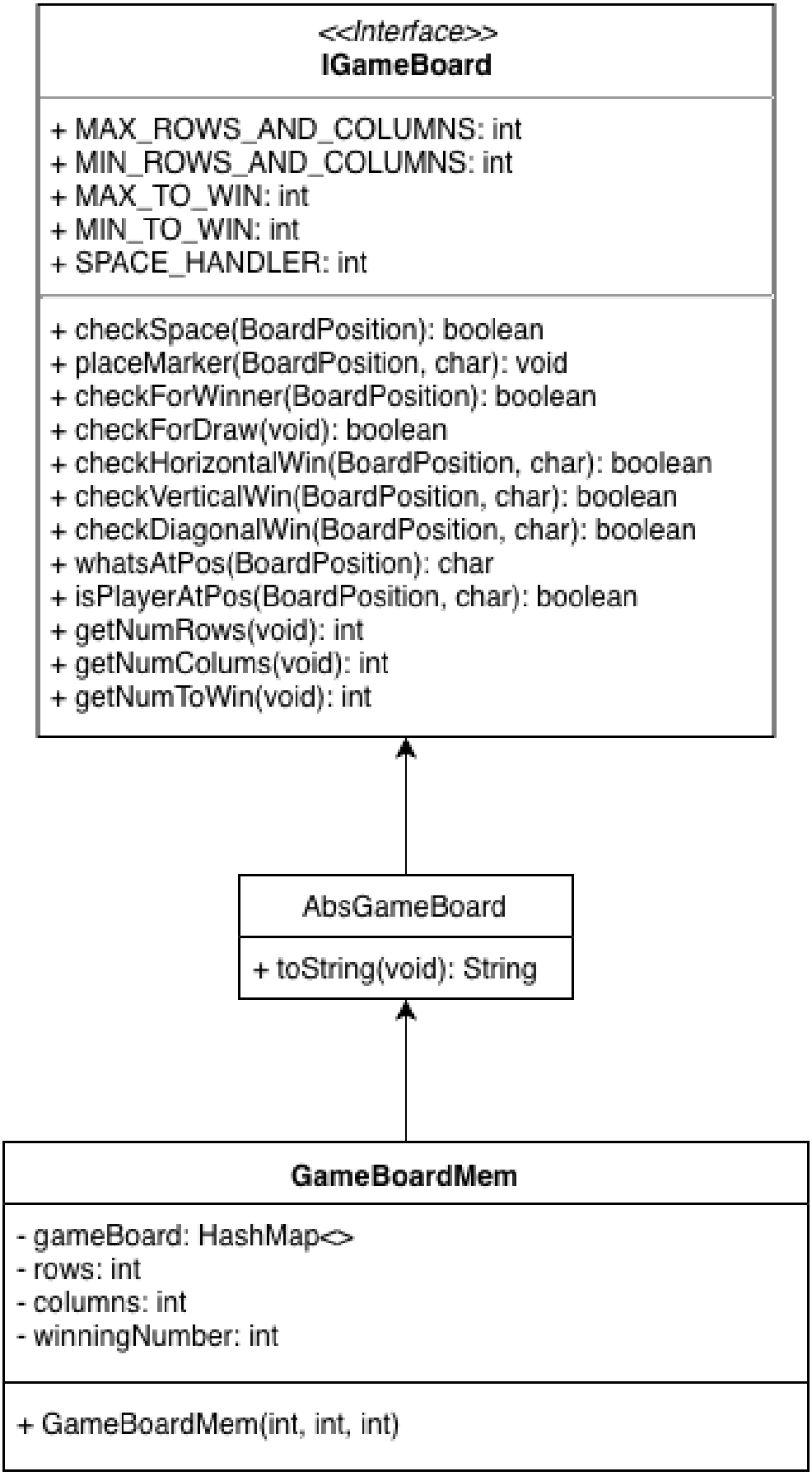

checkForDraw()
Create iterable at [0] [0] on game board
Is the char not a space?
no
Return false
yes
Iterate to next position
no
At the end?
yes
Return true


checkDiagonalWin()
Initialize player space counter to 1 and iterator equal to the passed BoardPosition
Is player space counter equal to winningNumber or is the next position down and to the right not occupied with current player or is the player space counter out of bounds?
yes
no
Increment x position, decrement y position and increment space counter
set iterator to original BoardPosition
Is player space counter equal to winningNumber or is the next position up and to left not occupied with the current playe or is the player space counter out of bounds r?
yes
no
Decrement x position, increment y position and increment space counter
set iterator to original BoardPosition
Is player space counter equal to winningNumber or is the next position up and to the right not occupied with current player or is the player space counter out of bounds ?
yes
no
Increment x position, increment y position and increment space counter
set iterator to original BoardPosition
Is player space counter equal to winningNumber or is the next position down and to left not occupied with the current player or is the player space counter out of bounds ?
yes
no
Decrement x position, decrement y position and increment space counter
Is player space counter winningNumber ?
no
Return false
yes
Return true

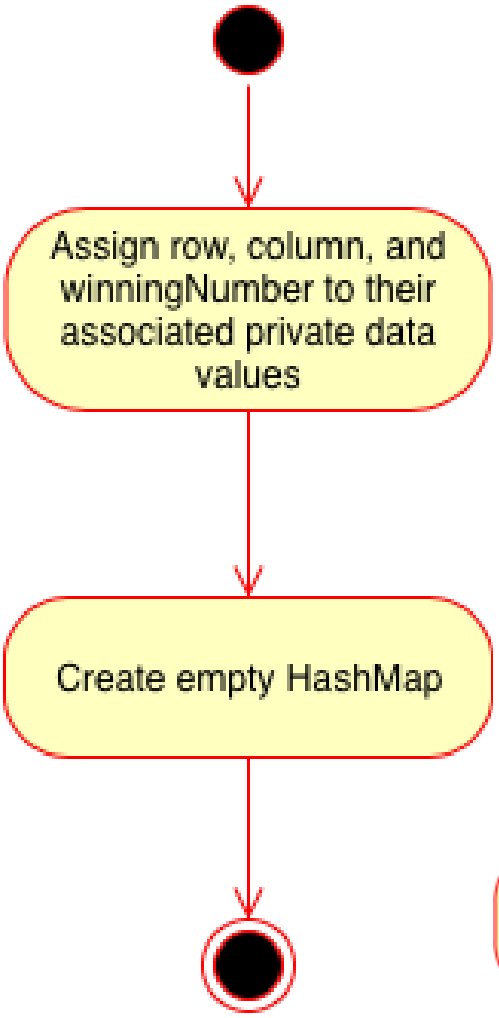

whatsAtPos()
Return the character at the location of the BoardPosition's X and Y


isPlayerAtPos()
Is the character that is at the location of the board position's x and y equal to the passed character?
no
Return false
yes
Return true

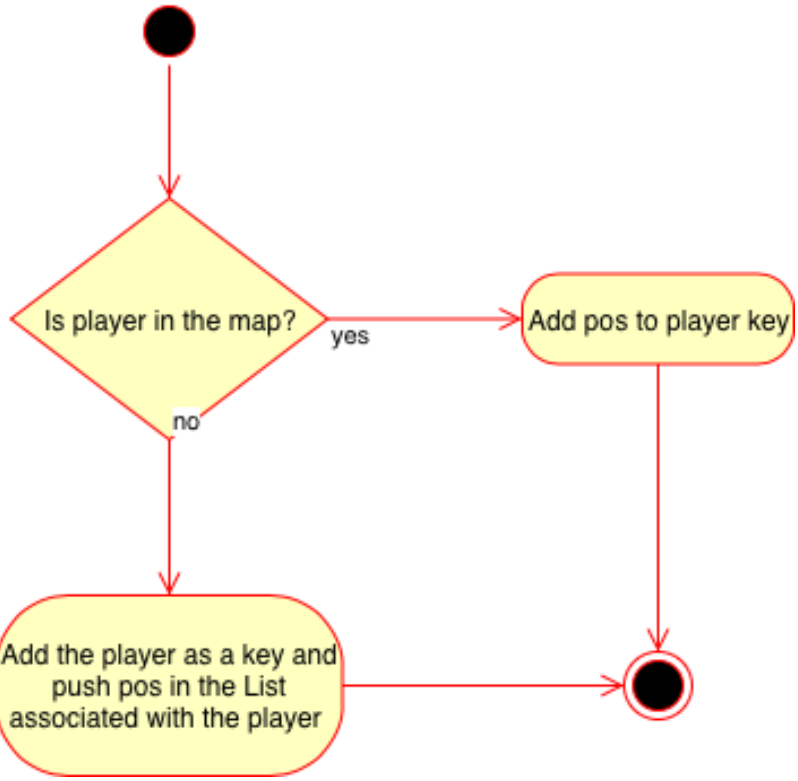

toString()
Create new string to hold the game board
Is row_iter >= getNumColumns?
yes
no
Append the row_iter and a 'l' character to the string
Append '\n'
Is row_iter >= getNumRows?
yes
no
Is row_iter less than SPACE_HANDLER?
no
yes
Append a space
Increment row_iter
Is col_iter >= getNumColumns?
yes
no
Return board string
Append whatsAtPos(BoardPosition(row_iter, col_iter)) and 'l'
Increment col_iter


<<Interface>>
IGameBoard
+ MAX_ROWS_AND_COLUMNS: int
+ MIN_ROWS_AND_COLUMNS: int
+ MAX_TO_WIN: int
+ MIN_TO_WIN: int
+ SPACE_HANDLER: int
+ checkSpace(BoardPosition): boolean
+ placeMarker(BoardPosition, char): void
+ checkForWinner(BoardPosition): boolean
+ checkForDraw(void): boolean
+ checkHorizontalWin(BoardPosition, char): boolean
+ checkVerticalWin(BoardPosition, char): boolean
+ checkDiagonalWin(BoardPosition, char): boolean
+ whatsAtPos(BoardPosition): char
+ isPlayerAtPos(BoardPosition, char): boolean
+ getNumRows(void): int
+ getNumColums(void): int
+ getNumToWin(void): int
AbsGameBoard
+ toString(void): String
GameBoardMem
- gameBoard: HashMap<>
- rows: int
- columns: int
- winningNumber: int
+ GameBoardMem(int, int, int)

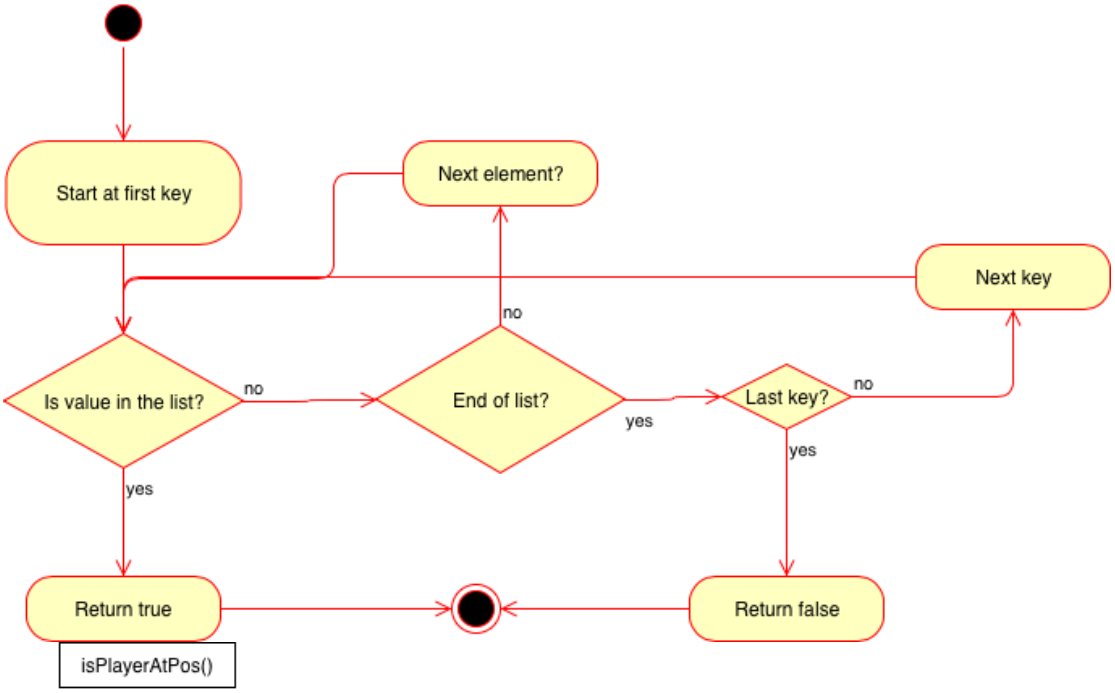
GameBoardMem()



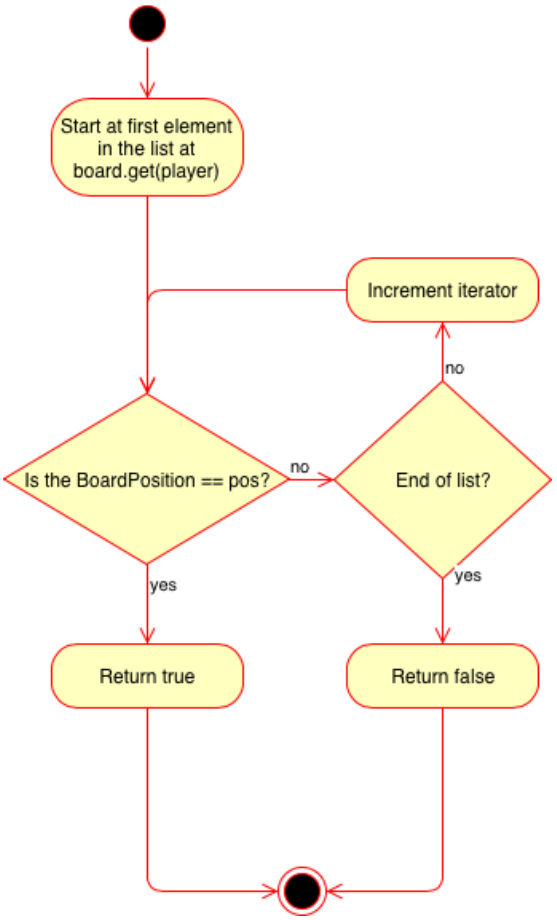
placeMarker()



whatsAtPos()



isPlayerAtPos()



Deployment

Included in the project is a makefile with the following targets:

- **default: compiles all the code. Runs with the *make* command**
- **run: runs the code. Runs with the command *make run***
- **clean: removes all compiled files from the package. Runs with the command *make clean***

In order to run Extended Tic Tac Toe, do the following:

1. **Make your way to the directory that contains the makefile and the package cpsc2150.extendedTicTacToe in the command line**
2. **When you are in the directory with the makefile, type the command *make***
3. **Type *make run***
4. **When the game is done, type *make clean***