Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

Homework #3 Report

Requirements Analysis

Functional Requirements

- As a user, I can place tokens into columns of a matrix to win connect4
- As a user, I can choose yes or no to decide if I want to play another game of connect4
- As a user, I can choose what kind of token I want to be, to play connect4
- As a user, I can decide how many columns the game board has, to play connect4
- As a user, I can decide how many rows the game board has, to play connect4
- As a user, I can decide how many tokens must be in a row, to win connect 4
- As a user, I can specify how many players can play a game of connect4
- As a user, I can decide if I want to play a fast, or memory efficient game of connect4

Non-functional Requirements

- The system must be coded in Java
- The system must be coded on the IntelliJ IDE
- The system must run on Clemson's SOC Unix machines
- The system must utilize a 2D array and a Map
- The system must print from the main function utilizing a string
- The system must be operationally ready by Sunday, 28 October 2018 @ 11:59 pm
- The system must always start with player 1
- The system must alternate turns between players
- The system must show the players the current board each turn
- The system must ask the appropriate player to select a column to add their token to
- The system must check if the last token placed results in a win
- The system must check if the last token placed results in a tie
- The system must prompt the users after a win or tie if they want to play again
- The system must show the gameboard to the users after a tie
- The system must show the gameboard to the users after a win
- The system must prompt the user to reenter a value of if the column chosen is full
- The system must prompt the user to reenter a value if their original value is invalid

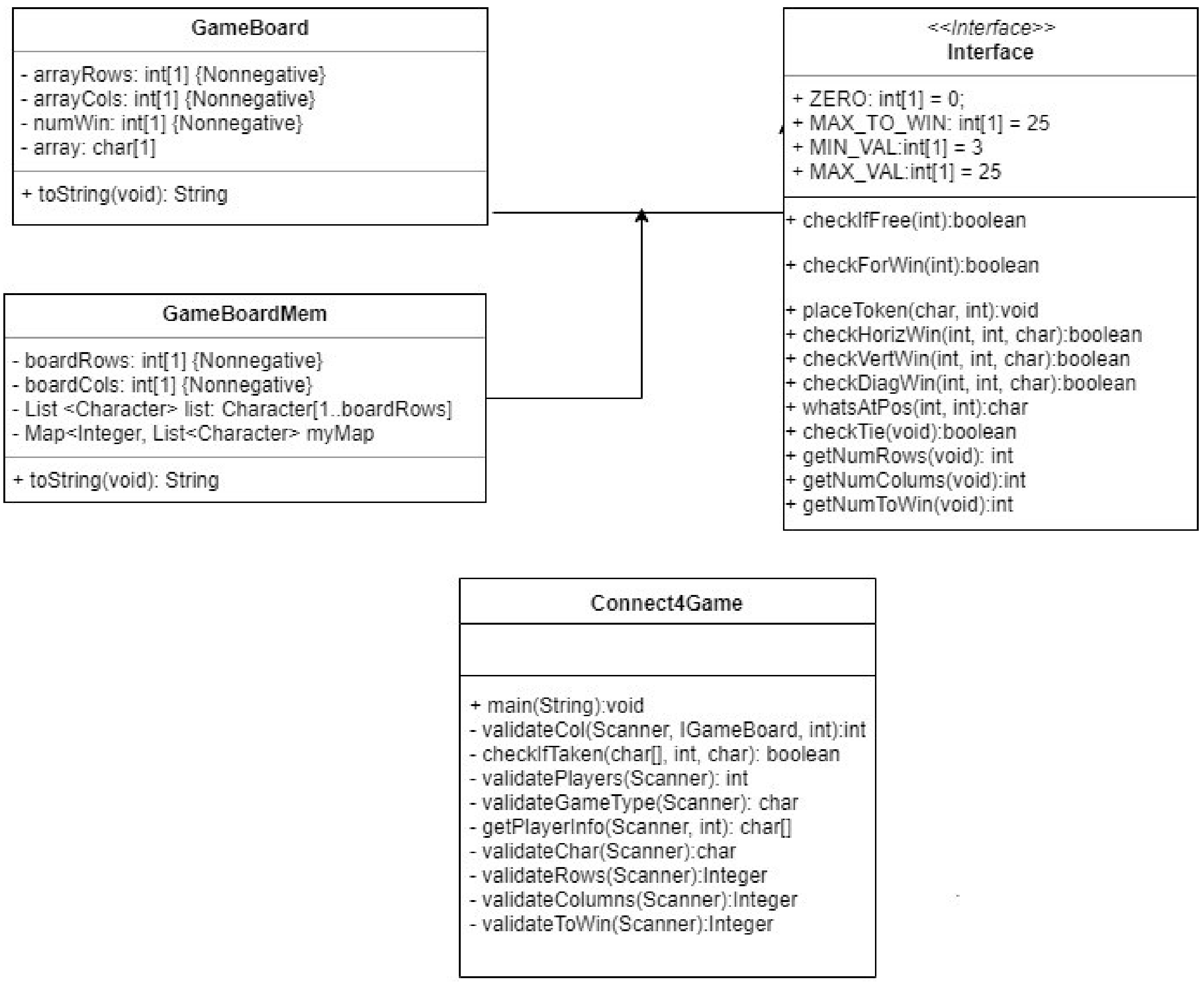
Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

- The system must have a Connect4Game class containing the main function
- The system must have an IGameBoard interface
- The system must have a GameBoard class implementing a 2D array
- The system must have a GameBoardMem class a map
- The system must code to the interface
- All methods (except for the main) must have preconditions and post-conditions in the Javadoc contracts
- All methods (except for main) must have the params and returns specified in the Javadoc comments
- The GameBoard class must have invariants specified at the top of the class file in Javadoc comments
- The GameBoardMem class must have invariants specified at the top of the class file in Javadoc comments
- The system must include a makefile which has make, run, and clean commands

Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

Design

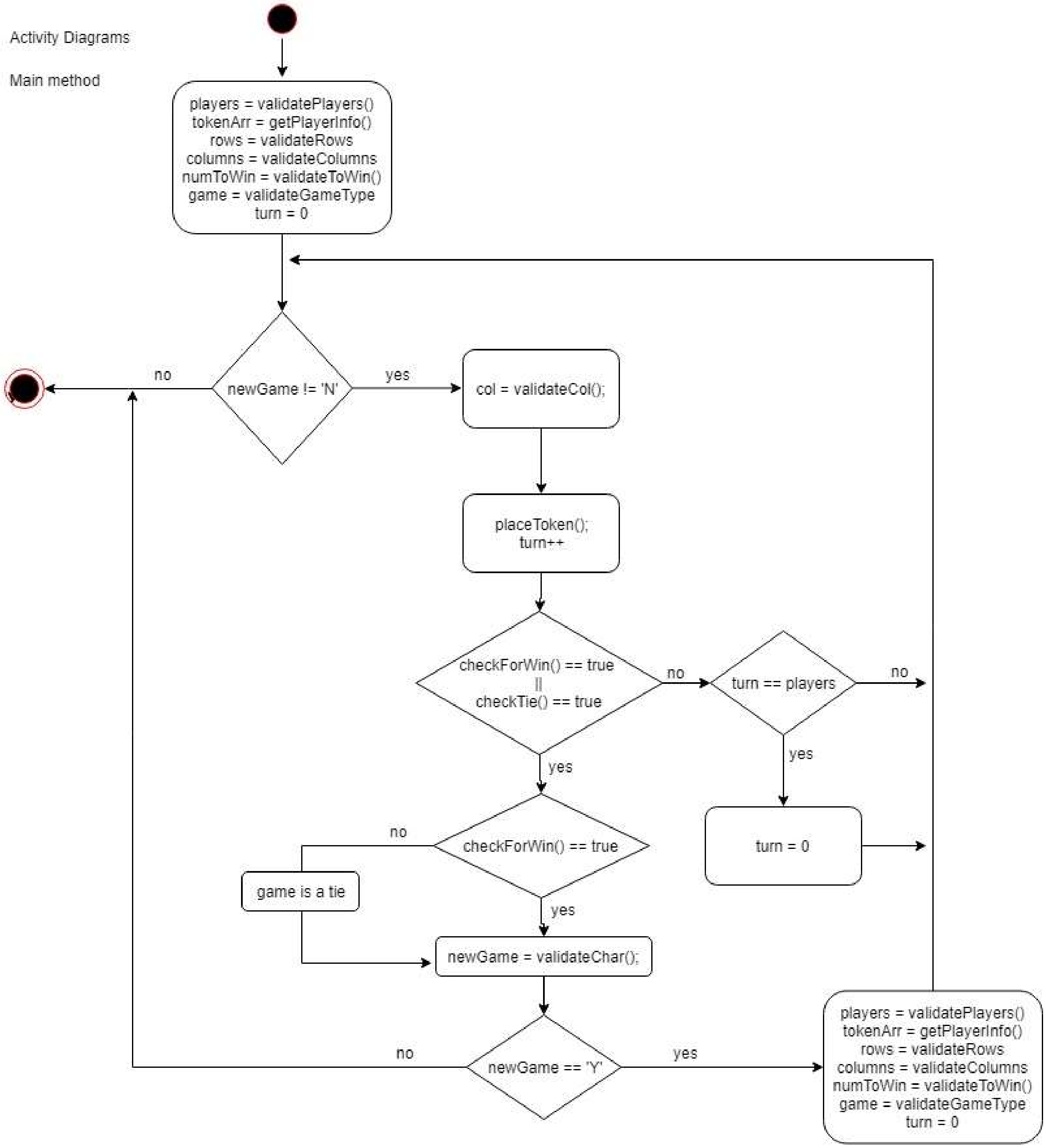
Class Diagrams


GameBoard
- arrayRows: int[1] {Nonnegative}
- arrayCols: int[1] {Nonnegative}
- numWin: int[1] {Nonnegative}
- array: char[1]
+ toString(void): String
<<Interface>>
Interface
+ ZERO: int[1] = 0;
+ MAX_TO_WIN: int[1] = 25
+ MIN_VAL:int[1] = 3
+ MAX_VAL:int[1] = 25
+ checkIfFree(int):boolean
+ checkForWin(int):boolean
+ placeToken(char, int):void
+ checkHorizWin(int, int, char):boolean
+ checkVertWin(int, int, char):boolean
+ checkDiagWin(int, int, char):boolean
+ whatsAtPos(int, int):char
+ checkTie(void):boolean
+ getNumRows(void): int
+ getNumColums(void):int
+ getNumToWin(void):int
GameBoardMem
- boardRows: int[1] {Nonnegative}
- boardCols: int[1] {Nonnegative}
- List <Character> list: Character[1..boardRows]
- Map<Integer, List<Character> myMap
+ toString(void): String
Connect4Game
+ main(String):void
- validateCol(Scanner, IGameBoard, int):int
- checkIfTaken(char[], int, char): boolean
- validatePlayers(Scanner): int
- validateGameType(Scanner): char
- getPlayerInfo(Scanner, int): char[]
- validateChar(Scanner):char
- validateRows(Scanner):Integer
- validateColumns(Scanner):Integer
- validateToWin(Scanner):Integer

Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

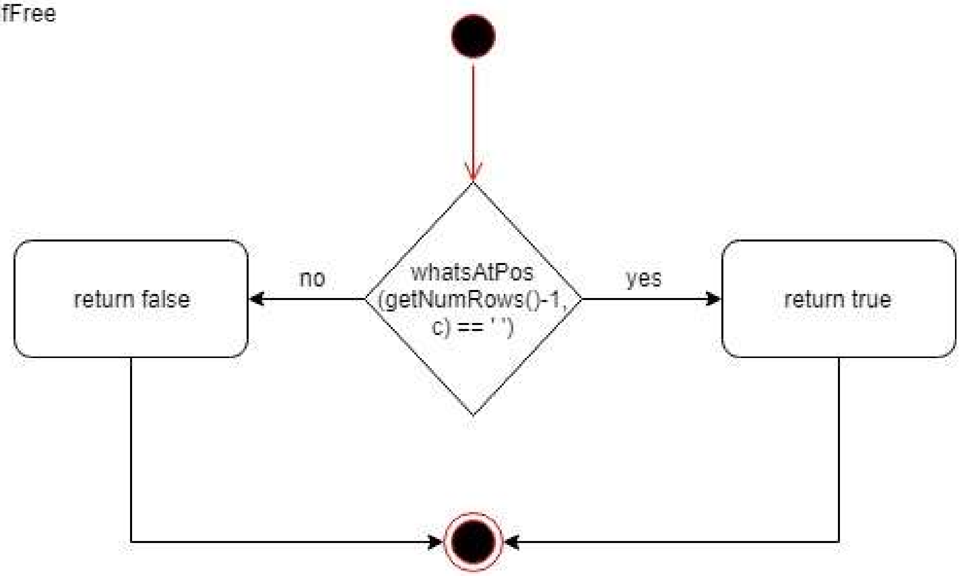
Activity Diagrams

Main method

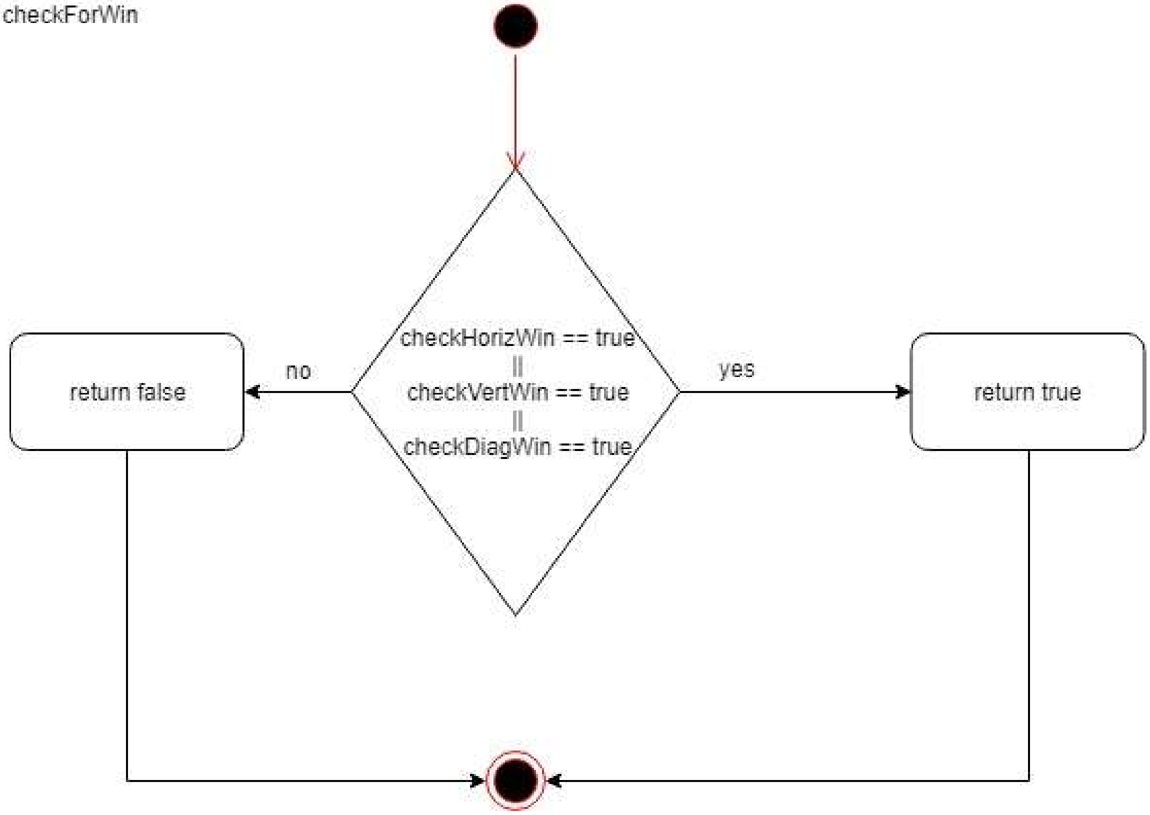


Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

default checkIfFree

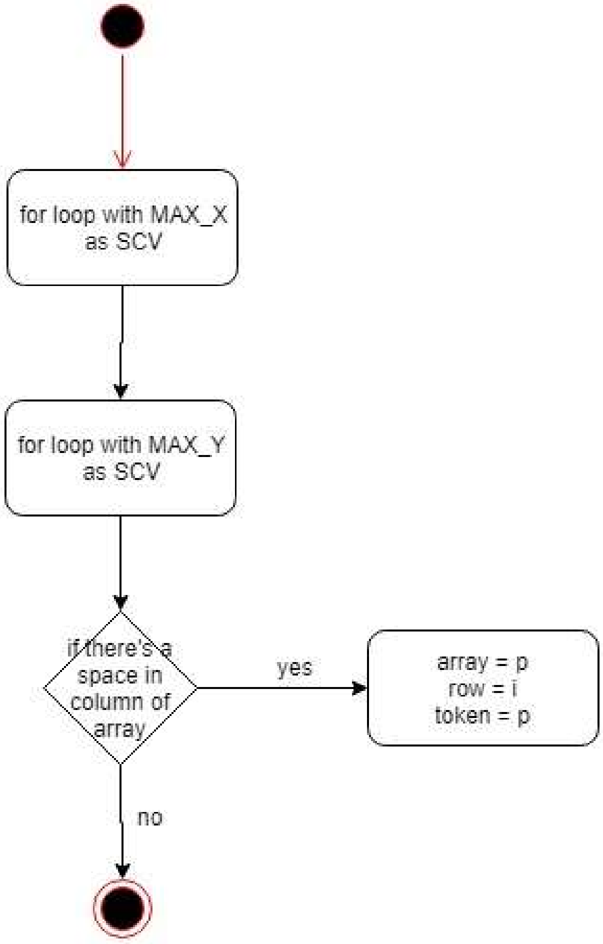


default checkForWin

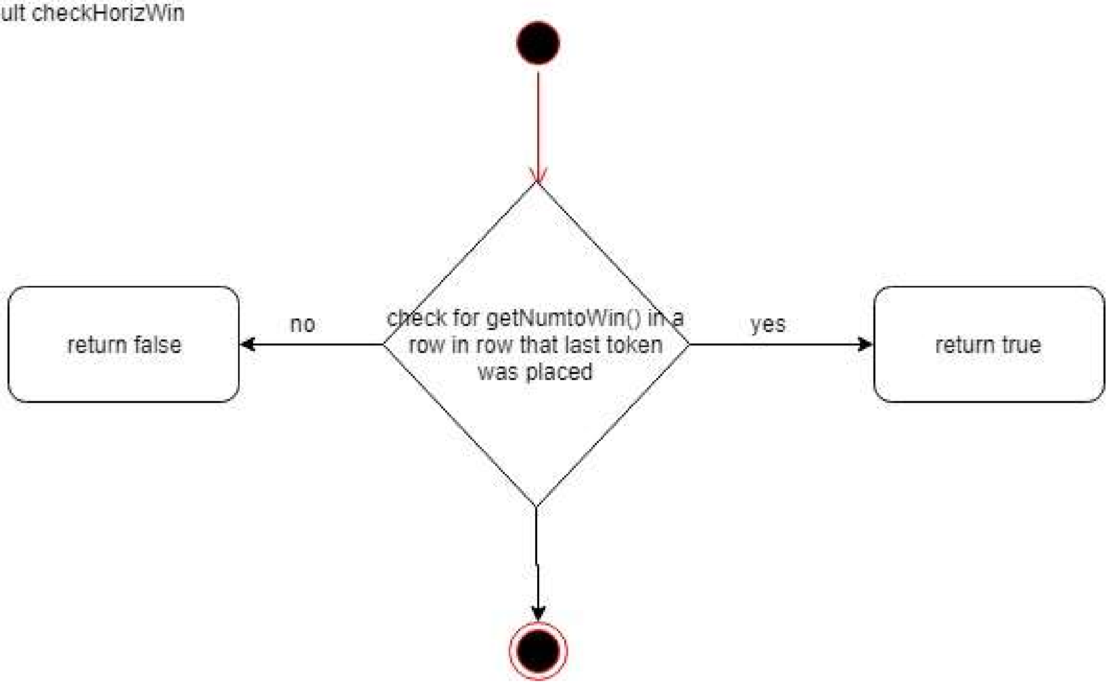


Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

placeToken

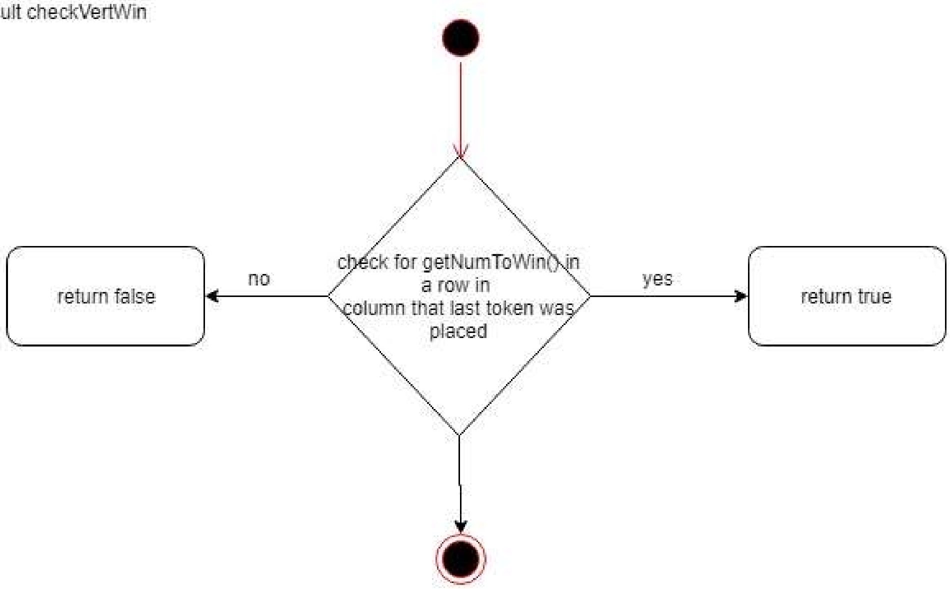

for loop with MAX_X as SCV
for loop with MAX_Y as SCV
if there's a space in column of array
yes
array = p
row = i
token = p
no

default checkHorizWin

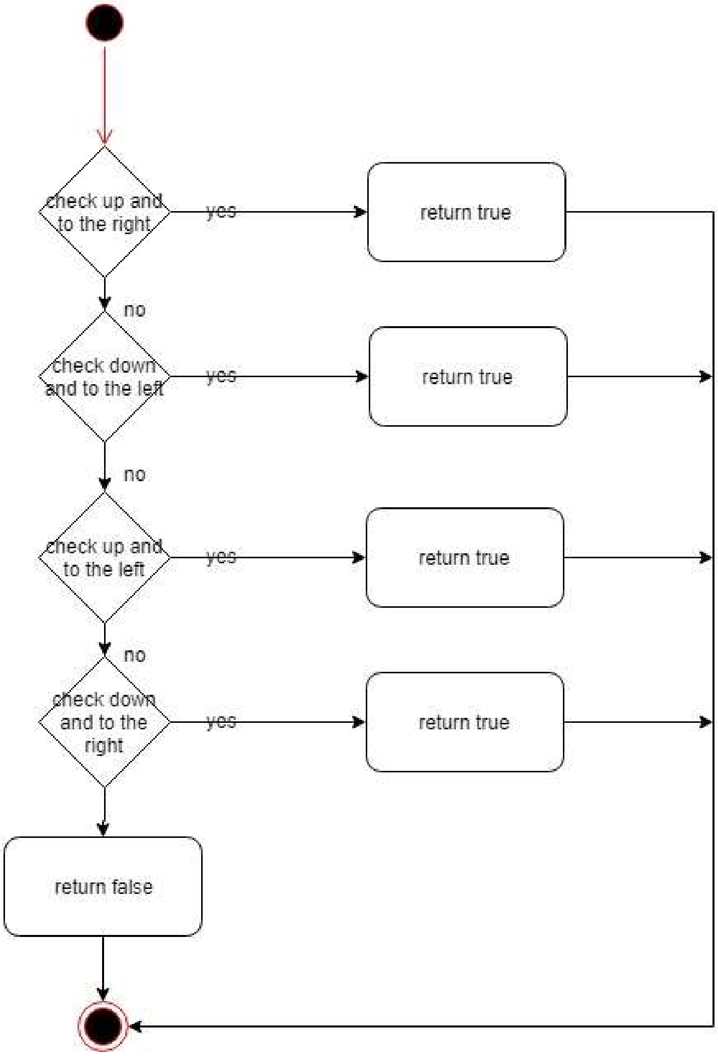

check for getNumtoWin() in a row in row that last token was placed
no
return false
yes
return true

Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

default checkVertWin

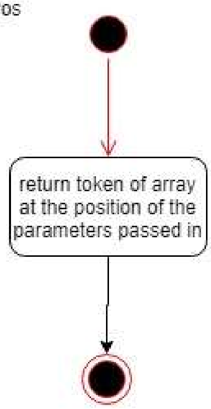

check for getNumToWin() in a row in column that last token was placed
no
return false
yes
return true

default checkDiagWin

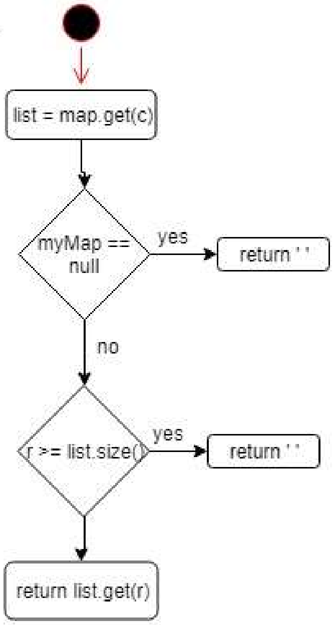


Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

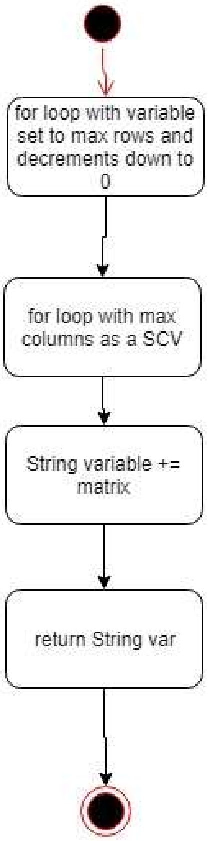
GameBoard whatsAtPos


return token of array at the position of the parameters passed in

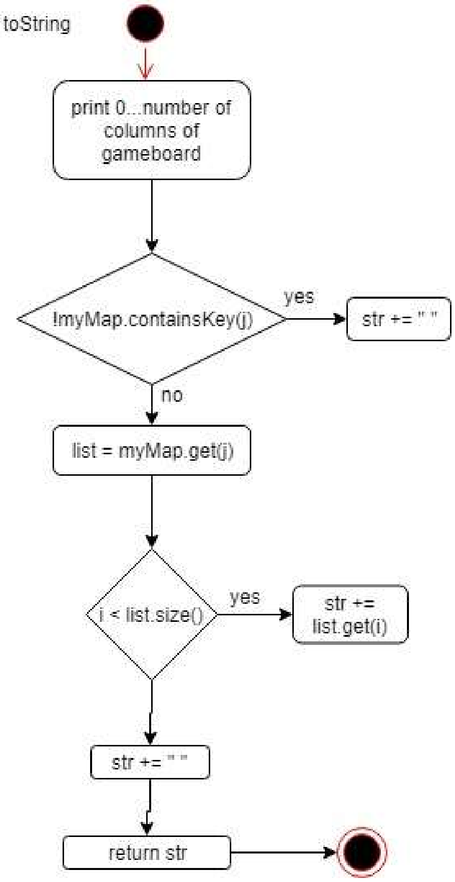
GameBoard whatsAtPos



GameBoard toString

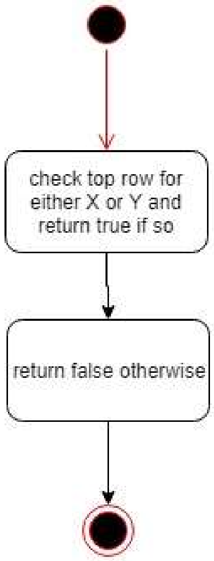

for loop with variable set to max rows and decrements down to 0
for loop with max columns as a SCV
String variable += matrix
return String var

GameBoardMem toString

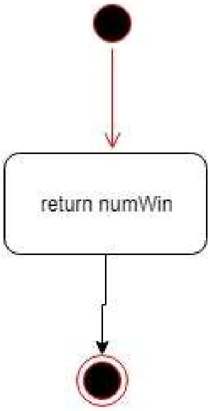


Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

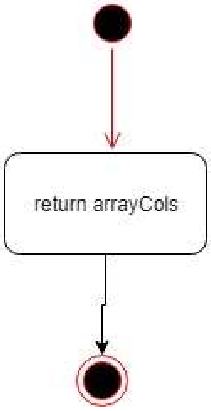
default checkTie



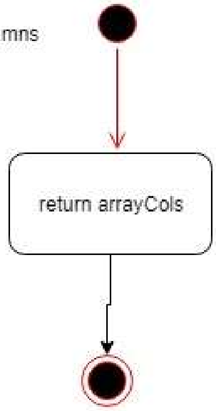
getNumToWin



getNumRows



getNumColumns



Wesley Lewis
CPSC 2150
Section 01
Homework #3
Due: 28 October 2018 @ 11:59 pm

Deployment

A makefile is included with this program. After unzipping the file and placing it onto a Unix machine, navigate to the directory in which the file was unzipped to. This is where the makefile is located. Once you have located the makefile type “make” to compile the program. Once the program has compiled, type “make run” to run the program. To remove the .class files, type “make clean.”