Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Homework #4 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

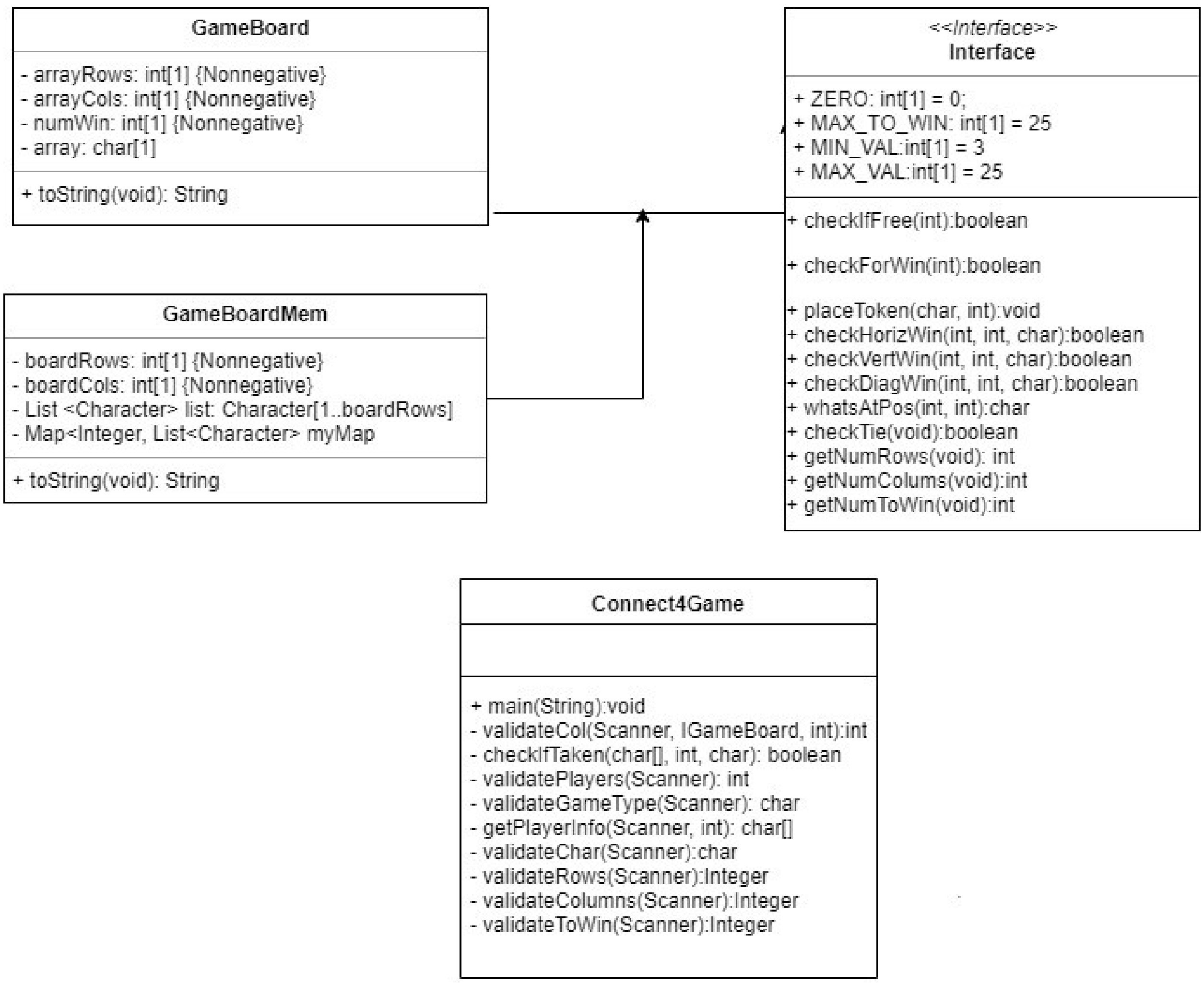
- The system must be coded in Java
- The system must be coded on the IntelliJ IDE
- The system must run on Clemson's SOC Unix machines
- The system must utilize a 2D array and a Map
- The system must print from the main function utilizing a string
- The system must be operationally ready by Sunday, 28 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

- 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 #4
Due: 18 November 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

GameBoardMem

- boardRows: int[1] {Nonnegative}
- boardCols: int[1] {Nonnegative}
- List <Character> list: Character[1..boardRows]
- Map<Integer, List<Character> myMap

+ 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

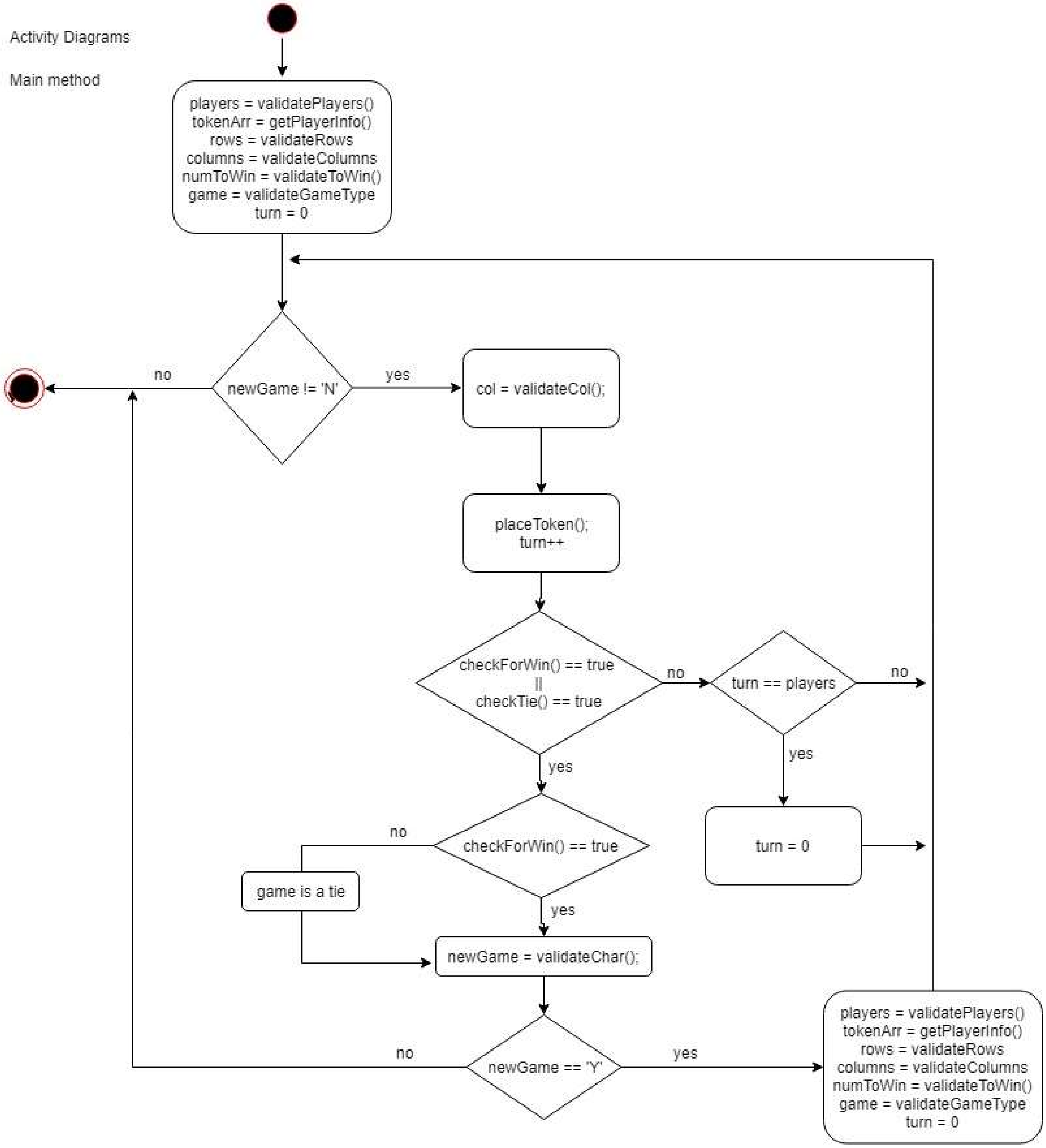
Connect4Game

+ 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 #4
Due: 18 November 2018 @ 11:59 pm

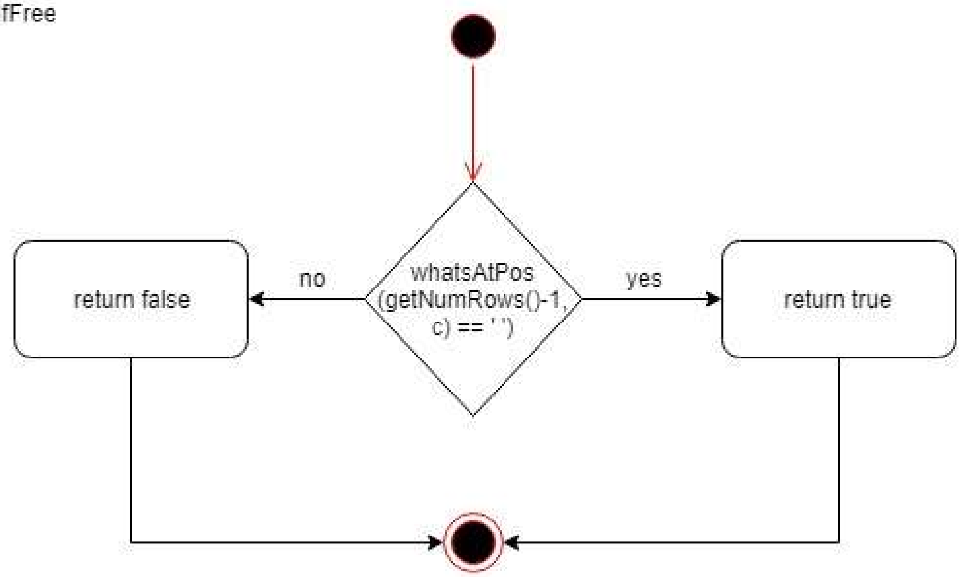
Activity Diagrams

Main method

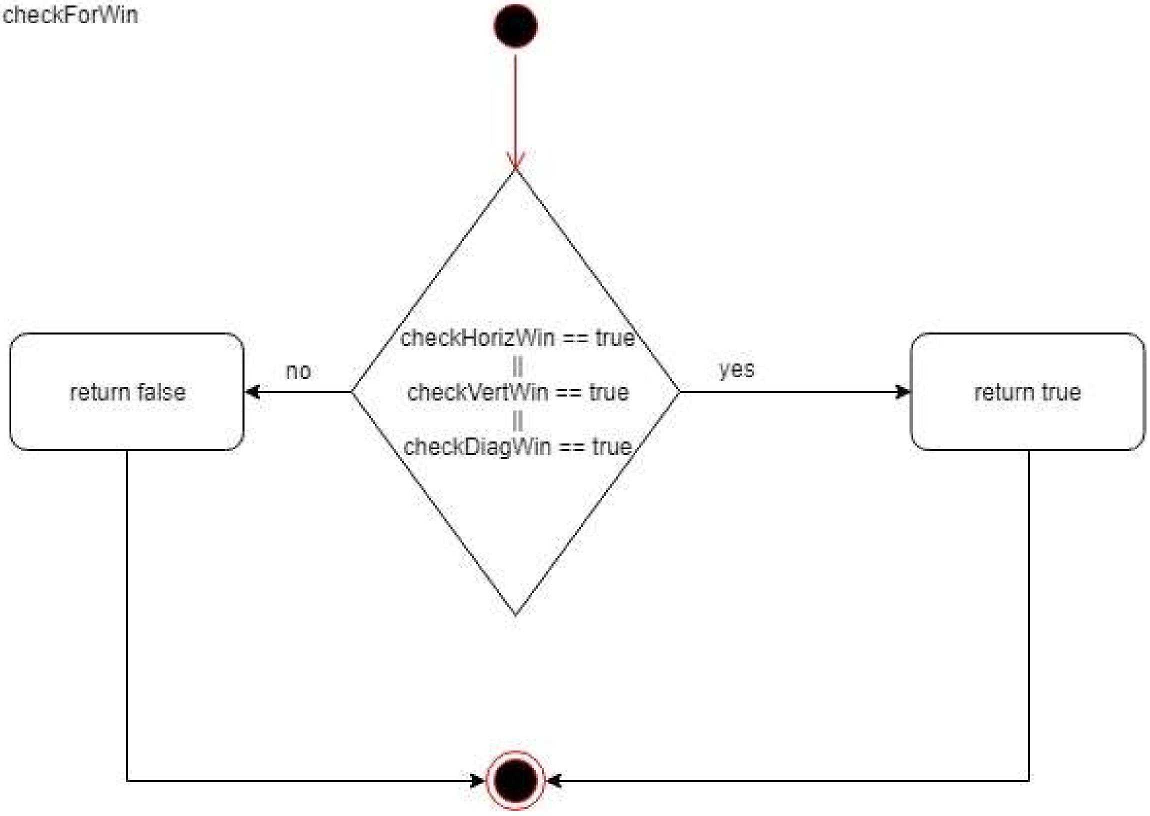


Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

default checkIfFree

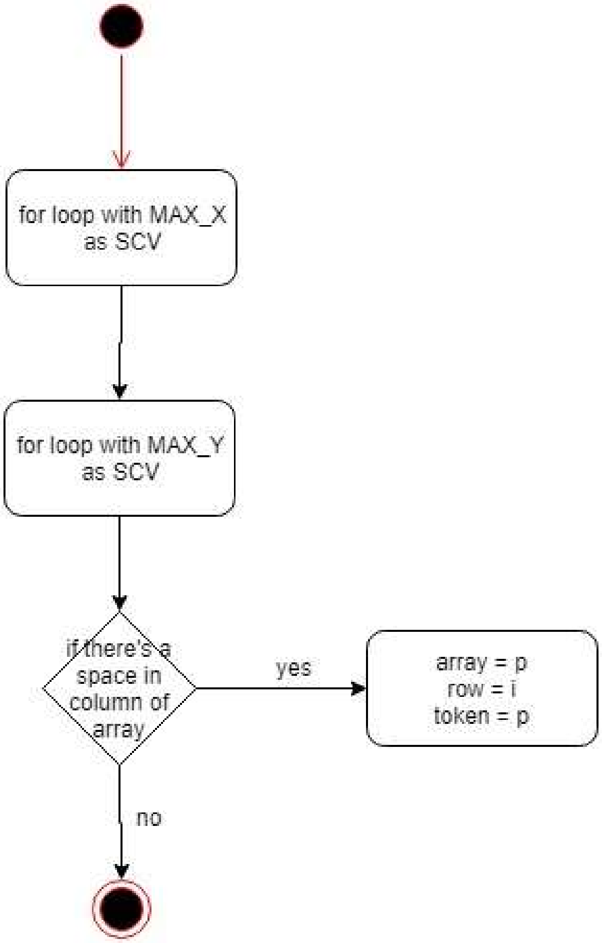


default checkForWin

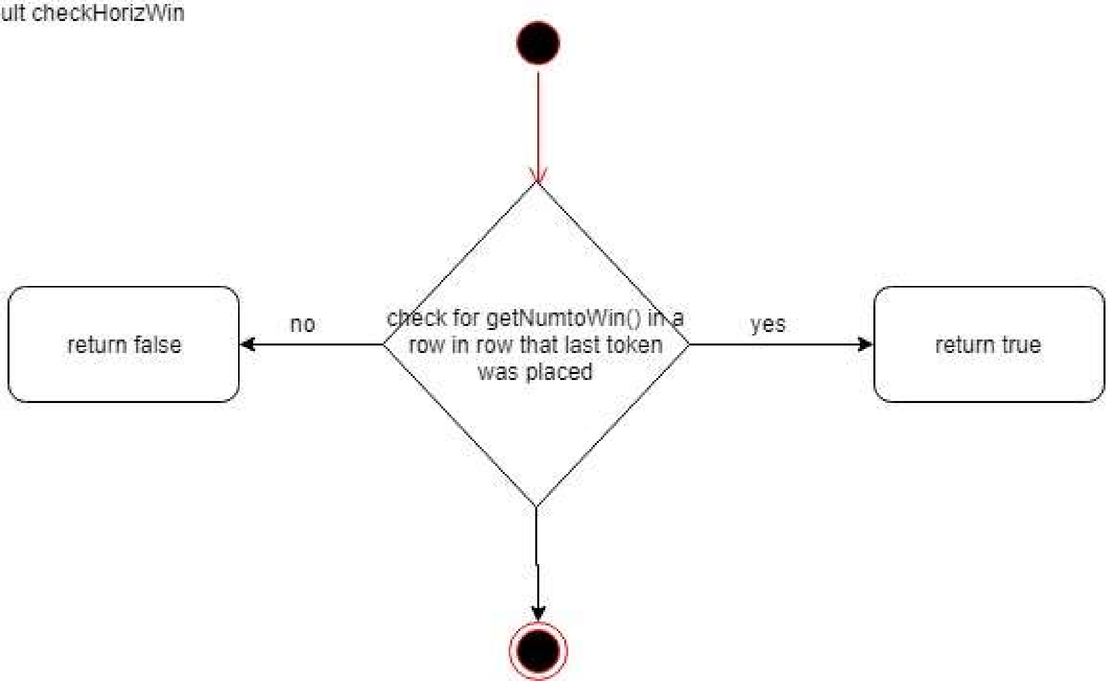


Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

placeToken

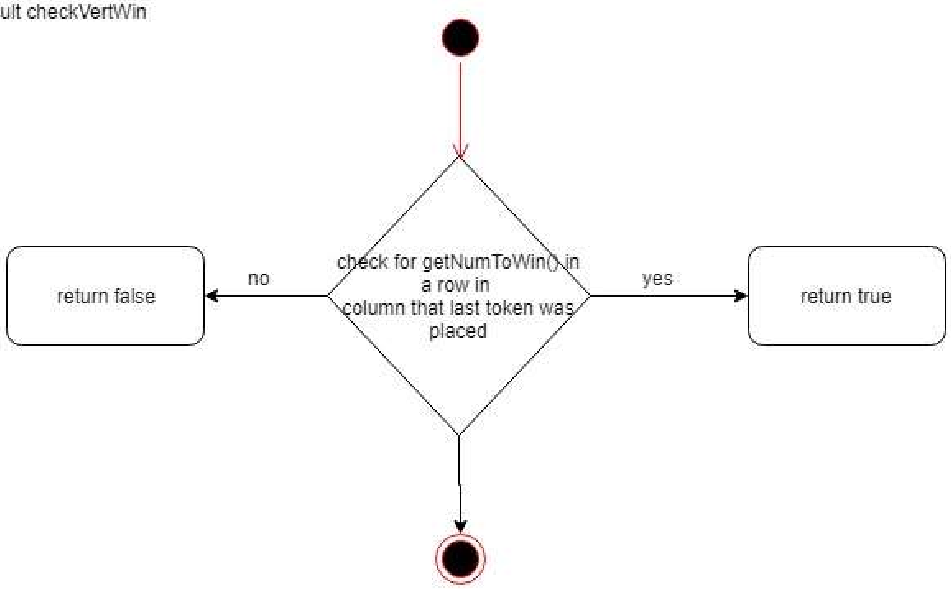

for loop with MAX_X as SCV
for loop with MAX_Y as SCV
if there's a space in column of array
yes
array = p
row = i
token = p
no

default checkHorizWin

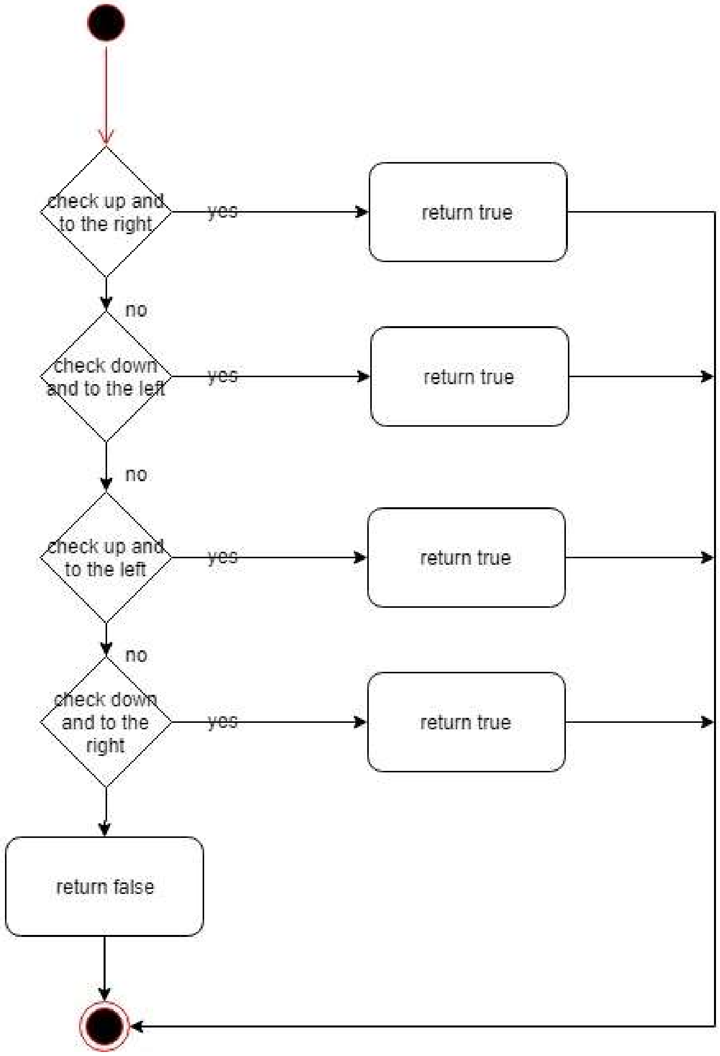

return false
no
check for getNumtoWin() in a row in row that last token was placed
yes
return true

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

default checkVertWin

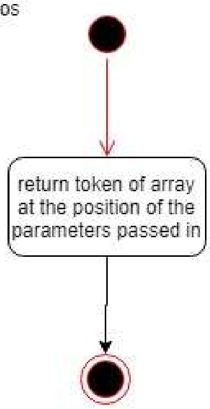

check for getNumToWin() in a row in column that last token was placed
no
return false
yes
return true

default checkDiagWin

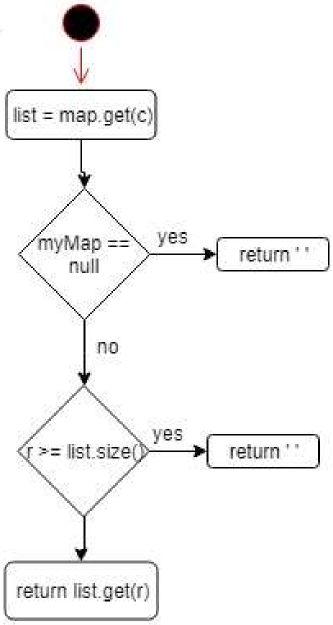


Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

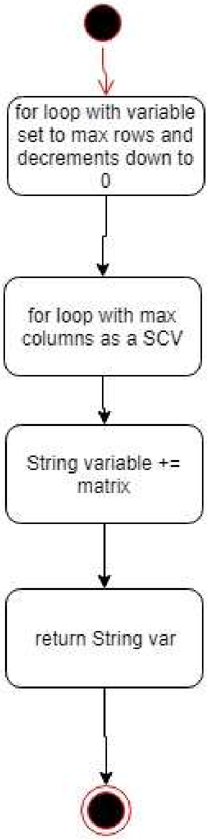
GameBoard whatsAtPos


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

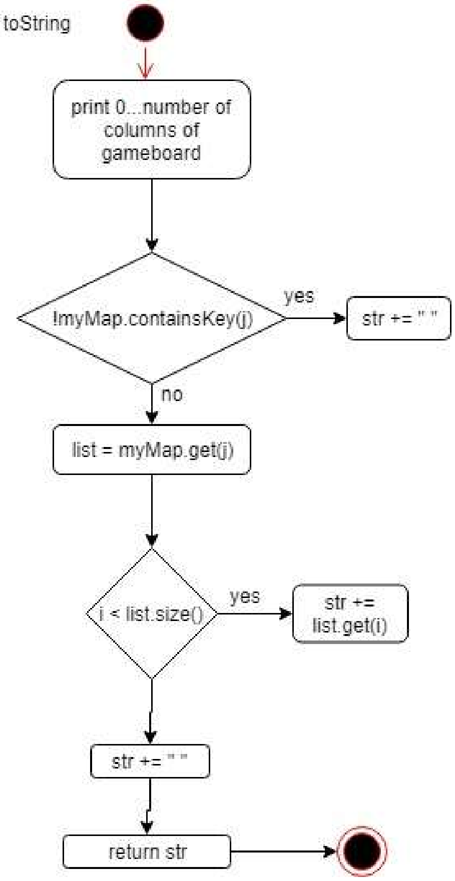
GameBoard whatsAtPos



GameBoard toString

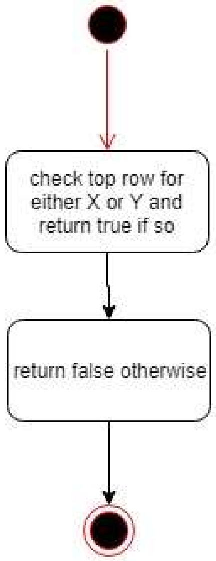

for loop with variable set to max rows and decrements down to 0
for loop with max columns as a SCV
String variable += matrix
return String var

GameBoardMem toString

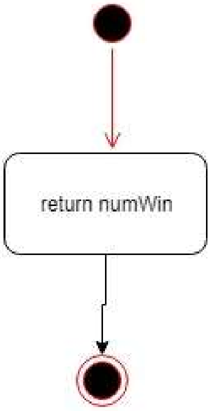


Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

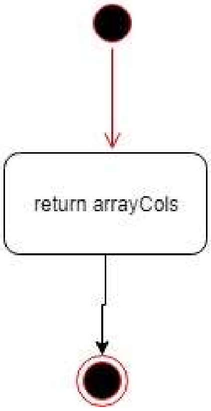
default checkTie



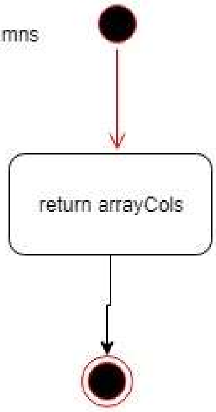
getNumToWin



getNumRows



getNumColumns



Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Testing

Constructor (int cols, int rows, int numToWin) case 1

Input	Output:	Reason:

cols = 3
rows = 3
numToWin = 3
	Gameboard with 3x3 rows/cols and number in a row needed to win of 3	This test case is unique and distinct because it tests that the constructor will "construct" the proper sized gameboard using the minimum rows/columns and number in a row to win
		Function:
		`testConstructor_min_cols_rows_numToWin`

Constructor (int cols, int rows, int numToWin) case 2

Input	Output:	Reason:

cols = 100
rows = 100
numToWin = 25
	Gameboard with 100x100 rows/cols and number in a row needed to win of 6	This test case is unique and distinct because it tests that the constructor will "construct" the proper sized gameboard using the maximum rows/columns and number in a row to win
		Function:
		`testConstructor_max_cols_rows_numToWin`

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Constructor (int cols, int rows, int numToWin) case 3

Input	Output:	Reason:

cols = 10
rows = 25
numToWin = 6
	Gameboard with 25x10 rows/cols and number in a row needed to win of 6	This test case is unique and distinct because it tests that the constructor will "construct" the proper sized gameboard using a random number of rows/columns and number in a row to win
		Function:
		`testConstructor_random_cols_rows_numToWin`

Boolean CheckIfFree (int c) case 1

Input	Output:	Reason:
c = 1	Return true	This test case is unique and distinct because if there is an empty space in column one when the gameboard has minimum rows/cols Function: `testCheckIfFree_true_min`

Boolean CheckIfFree (int c) case 2

Input	Output:	Reason:
c = 1 (board too large to put into table)	Return true	This test case is unique and distinct because Function: `testCheckIfFree_true_max`

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr>
<td>Input
c = 1
<table><tr><td></td><td>X</td><td></td></tr><tr><td></td><td>X</td><td></td></tr><tr><td></td><td>X</td><td></td></tr></table></td>
<td>Output:
Return false</td>
<td>Reason:
This test case is unique and distinct because it returns false when the column is full of tokens for column one
Function:
testCheckIfFree_false</td>
</tr>
</table>

Boolean CheckHorizWin (int r, int c, char p) case 1

<table>
<tr>
<td>Input
State: (number to win = 3)
c = 2
r = 0
p = x
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>Output:
checkHorizWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because the last x was placed at the end of the string of 3 consecutive x's resulting in a win
Function:
testCheckHorizWin_true_last_marker</td>
</tr>
</table>

Boolean CheckHorizWin (int r, int c, char p) case 2

<table>
<tr>
<td>Input
State: (number to win = 3)
c = 1
r = 0
p = x
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td>X</td><td>X</td><td></td></tr></table></td>
<td>checkHorizWin = false
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because the last x was placed resulting in a return of false because there are not 3 tokens in a row horizontally
Function:
testCheckHorizWin_false</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Boolean CheckHorizWin (int r, int c, char p) case 3

<table>
<tr>
<td>Input
State: (number to win = 3)

c = 1
r = 0
p = x

<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>Output:

checkHorizWin = true

state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because the last x was placed in the middle of the string of 3 consecutive x’s resulting in a win

Function:

testCheckHorizWin_true_mid</td>
</tr>
</table>

Boolean CheckHorizWin (int r, int c, char p) case 4

<table>
<tr>
<td>Input
State: (number to win = 3)

c = 0
r = 0
p = x

<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>Output:

checkHorizWin = true

state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because the last x was placed in the beginning of the string of 3 consecutive x’s resulting in a win

Function:

testCheckHorizWin_true_left</td>
</tr>
</table>

Boolean CheckHorizWin (int r, int c, char p) case 5

<table>
<tr>
<td>Input

c = 0
r = 5
p = O

<table><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>X</td><td>O</td><td>X</td><td>O</td><td>X</td><td>O</td><td></td><td></td></tr></table></td>
<td>Output:

checkHorizWin = false

state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because the O was placed at the end of alternating tokens and checks that the function will return false as it should

Function:

testCheckHorizWin_false_alternating</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Boolean CheckVertWin (int r, int c, char p) case 1

<table>
<tr>
<td>Input
c = 0
r = 2
p = X
<table><tr><td>X</td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr></table></td>
<td>Output:
checkVertWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests if the function will return true using the minimum number of required rows, columns and number in a row to win
Function:
testCheckVertWin_true_min_num_win</td>
</tr>
</table>

Boolean CheckVertWin (int r, int c, char p) case 2

<table>
<tr>
<td>Input
c = 0
r = 1
p = X
<table><tr><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr></table></td>
<td>Output:
checkVertWin = false
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests if the function will return false using the minimum number of required rows, columns and number in a row to win
Function:
testCheckVertWin_false_min_num_win</td>
</tr>
</table>

Boolean CheckVertWin (int r, int c, char p) case 3

<table>
<tr>
<td>Input
c = 2
r = 25
p = X
(gameboard too big to fit in table)</td>
<td>Output:
checkVertWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests if the function will return true using the max number of required rows, columns and number in a row to win
Function:
testCheckVertWin_true_max_num_win</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Input	Output:	Reason:

c = 2
r = 22
p = X
(gameboard too big to fit in table)
	checkVertWin = false
	state of the board is unchanged
		This test case is unique and distinct because it tests if the function will return false using the max number of required rows, columns and number in a row to win
		Function:
		`testCheckVertWin_false_max_num_win`

Boolean CheckVertWin (int r, int c, char p) case 5

Input	Output:	Reason:

c = 2
r = 24
p = X
(gameboard too big to fit in table)
	checkVertWin = true
	state of the board is unchanged
		This test case is unique and distinct because it tests for a true return when a win condition is met at the top of a column
		Function:
		`testCheckVertWin_true_top`

Boolean CheckDiagWin (int c, int r, char p) case 1

<table>
<tr><th>Input</th><th>Output:</th><th>Reason:</th></tr>
<tr>
<td>c = 4
r = 3
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>X</td><td></td></tr>
<tr><td></td><td></td><td></td><td>X</td><td>O</td><td></td></tr>
<tr><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>O</td><td></td></tr>
</table>
</td>
<td>checkDiagWin = true
state of the board is unchanged</td>
<td>This test case is unique and distinct because it tests that a win condition is met when a token is placed at the end of a consecutive string of 4 up and to the right diagonally
Function:
testCheckDiagWin_up_and_right_top</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr>
<td>Input
c = 1
r = 0
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>X</td><td></td></tr>
<tr><td></td><td></td><td></td><td>X</td><td>O</td><td></td></tr>
<tr><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>O</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that a win condition is met when a token is placed at the beginning of a consecutive string of 4 up and to the right diagonally
Function:
testCheckDiagWin_up_and_right_bot</td>
</tr>
</table>

Boolean CheckDiagWin (int c, int r, char p) case 3

<table>
<tr>
<td>Input
c = 2
r = 1
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>X</td><td></td></tr>
<tr><td></td><td></td><td></td><td>X</td><td>O</td><td></td></tr>
<tr><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>O</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that a win condition is met when a token is placed in the middle of a consecutive string of 4 up and to the right diagonally
Function:
testCheckDiagWin_up_and_right_mid</td>
</tr>
</table>

Boolean CheckDiagWin (int c, int r, char p) case 4

<table>
<tr>
<td>Input
c = 1
r = 3
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>X</td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>X</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that a win condition is met when a token is placed at the end of a consecutive string of 4 up and to the left diagonally
Function:
testCheckDiagWin_up_and_left_top</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr>
<td>Input
c = 4
r = 0
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>X</td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>X</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that a win condition is met when a token is placed at the beginning of a consecutive string of 4 up and to the left diagonally
Function:
<code>testCheckDiagWin_up_and_left_bot</code></td>
</tr>
</table>

Boolean CheckDiagWin (int c, int r, char p) case 6

<table>
<tr>
<td>Input
c = 3
r = 0
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>X</td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>X</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = true
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that a win condition is met when a token is placed in the middle of a consecutive string of 4 up and to the left diagonally
Function:
<code>testCheckDiagWin_up_and_left_mid</code></td>
</tr>
</table>

Boolean CheckDiagWin (int c, int r, char p) case 7

<table>
<tr>
<td>Input
c = 3
r = 0
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>O</td><td>X</td><td></td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>X</td><td>X</td><td></td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>X</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = false
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests the function will return false when a win condition is not met up and to the left
Function:
<code>testConstructor_false_up_and_left</code></td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr>
<td>Input
c = 3
r = 2
p = X
<table>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>O</td><td></td></tr>
<tr><td></td><td></td><td></td><td>X</td><td>O</td><td></td></tr>
<tr><td></td><td></td><td>X</td><td>X</td><td>X</td><td></td></tr>
<tr><td></td><td>X</td><td>O</td><td>O</td><td>O</td><td></td></tr>
</table>
</td>
<td>Output:
checkDiagWin = false
state of the board is unchanged</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return false when a win condition is not met up and to the right
Function:
<code>testConstructor_false_up_and_right</code></td>
</tr>
</table>

CheckTie (void) case 1

<table>
<tr>
<td>Input
<table>
<tr><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td></tr>
<tr><td>X</td><td>X</td><td>X</td></tr>
</table>
</td>
<td>Output:
checkTie = true</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return true when the board is full of tokens with minimum rows and columns
Function:
<code>testCheckTie_true_min_rows_cols</code></td>
</tr>
</table>

CheckTie (void) case 2

<table>
<tr>
<td>Input
Borad has 100 rows, 100 columns
(board too big for table)</td>
<td>Output:
checkTie = true</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return true when the board is full of tokens with maximum rows and columns
Function:
<code>testCheckTie_true_max_rows_cols</code></td>
</tr>
</table>

<table>
<tr>
<td>Input
<table><tr><td>X</td><td>X</td><td></td></tr><tr><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>Output:
checkTie = false</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return false when the board is not full (one empty space) with a board of minimum rows/columns
Function:
testCheckTie_false_min_rows_cols</td>
</tr>
</table>

CheckTie (void) case 4

<table>
<tr>
<td>Input
Board has 100 rows, 100 columns
(board too big for table)= 3</td>
<td>Output:
checkTie = false</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return false when the board is not full (one empty space) with a board of maximum rows/columns
Function:
testCheckTie_false_max_rows_cols</td>
</tr>
</table>

Char WhatsAtPos (int r, int c) case 1

<table>
<tr>
<td>Input
r = 0
c = 0
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr></table></td>
<td>Output:
Returns ‘X’</td>
<td>Reason:
This test case is unique and distinct because it tests that the function will return the proper token at the bottom left of the game board
Function:
testWhatsAtPos_bot_left</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr><td>Input</td><td>Output:</td><td>Reason:</td></tr>
<tr><td>r = 0
c = 2
<table><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td>X</td></tr></table></td><td>Returns ‘X’</td><td>This test case is unique and distinct because it tests that the function will return the proper token at the bottom right of the game board
Function:
<code>testWhatsAtPos_bot_right</code></td></tr>
</table>

Char WhatsAtPos (int r, int c) case 3

<table>
<tr><td>Input</td><td>Output:</td><td>Reason:</td></tr>
<tr><td>r = 2
c = 2
<table><tr><td></td><td></td><td>X</td></tr><tr><td></td><td></td><td>X</td></tr><tr><td></td><td></td><td>X</td></tr></table></td><td>Returns ‘X’</td><td>This test case is unique and distinct because it tests that the function will return the proper token at the top right of the game board
Function:
<code>testWhatsAtPos_top_right</code></td></tr>
</table>

Char WhatsAtPos (int r, int c) case 4

<table>
<tr><td>Input</td><td>Output:</td><td>Reason:</td></tr>
<tr><td>r = 2
c = 2
<table><tr><td>X</td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr><tr><td>X</td><td></td><td></td></tr></table></td><td>Returns ‘X’</td><td>This test case is unique and distinct because it tests that the function will return the proper token at the top left of the game board
Function:
<code>testWhatsAtPos_top_left</code></td></tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

<table>
<tr>
<td>Input
<table><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr><tr><td>X</td><td>X</td><td>X</td><td>X</td></tr></table></td>
<td>Output:
Returns ‘X’</td>
<td>Reason:
This test case is unique and distinct because it tests every position of the gameboard testing that it will return X as the token in each position of the gameboard when it is full of X’s
Function:
<code>testWhatsAtPos_full</code></td>
</tr>
</table>

Char WhatsAtPos (int r, int c) case 6

<table>
<tr>
<td>Input
<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr></table></td>
<td>Output:
Returns ‘ ‘</td>
<td>Reason:
This test case is unique and distinct because it tests every position of the gameboard testing that it will return ‘ ‘ as the token in each position of the gameboard when it is empty
Function:
<code>testWhatsAtPos_empty</code></td>
</tr>
</table>

Char WhatsAtPos (int r, int c) case 7

<table>
<tr>
<td>Input
<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr></table></td>
<td>Output:
Returns ‘X’</td>
<td>Reason:
This test case is unique and distinct because it tests that it will return what is in the middle of the gameboard
Function:
<code>testWhatsAtPos_middle</code></td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Void PlaceToken (char p, int c) case 1

<table>
<tr>
<td>Input
p = ‘X’
c = 0</td>
<td>Output:<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td><td></td></tr></table></td>
<td>Reason:
This test case is unique and distinct because it tests that the function will place the token in the correct expected place in the game board. In this case, the bottom left
Function:
testPlaceToken_bot_left</td>
</tr>
</table>

Void PlaceToken (char p, int c) case 2

<table>
<tr>
<td>Input
p = ‘X’
c = 3</td>
<td>Output:<table><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>X</td></tr></table></td>
<td>Reason:
This test case is unique and distinct because it tests that the function will place the token in the correct expected place in the game board. In this case, the bottom right
Function:
testPlaceToken_bot_right</td>
</tr>
</table>

Void PlaceToken (char p, int c) case 3

<table>
<tr>
<td>Input
p = ‘X’
c = 3</td>
<td>Output:<table><tr><td>X</td><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td><td></td></tr><tr><td>X</td><td></td><td></td><td></td></tr></table></td>
<td>Reason:
This test case is unique and distinct because it tests that the function will place the token in the correct expected place in the game board. In this case, the top left
Function:
testPlaceToken_top_left</td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 2018 @ 11:59 pm

Void PlaceToken (char p, int c) case 4

<table>
<tr>
<td>Input
p = ‘X’
c = 4</td>
<td>Output:
<table><tr><td></td><td></td><td></td><td>X</td></tr><tr><td></td><td></td><td></td><td>X</td></tr><tr><td></td><td></td><td></td><td>X</td></tr><tr><td></td><td></td><td></td><td>X</td></tr></table></td>
<td>Reason:
This test case is unique and distinct because it tests that the function will place the token in the correct expected place in the game board. In this case, the top left
Function:
<code>testPlaceToken_top_right</code></td>
</tr>
</table>

Void PlaceToken (char p, int c) case 5

<table>
<tr>
<td>Input
p = ‘X’
c = 2</td>
<td>Output:
<table><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr><tr><td></td><td></td><td>X</td><td></td><td></td></tr></table></td>
<td>Reason:
This test case is unique and distinct because it tests that the function will place the token in the correct expected place in the game board. In this case, the middle
Function:
<code>testPlaceToken_middle</code></td>
</tr>
</table>

Wesley Lewis
CPSC 2150
Section 01
Homework #4
Due: 18 November 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. Type “make test” to compile the test cases. And type “make test” to run the test cases. To remove the .class files, type “make clean.”