Kellen Haas

CPSC 2150

Project 4

11/17/2020

Requirements Analysis

Functional Requirements

1. As a user, I can choose which row I want to place my marker in.
2. As a user, I can choose which column I want to place my marker in.
3. As a user, I can view the game board before my turn and after my turn with my updated marker that I just placed.
4. As a user, after the first user takes their turn, I can then choose my row.
5. As a user, after the first user takes their turn, I can choose my column.
6. As a user, I can expect that the system will notify me and my opponent if someone has won horizontally.
7. As a user, I can expect that the system will notify me and my opponent if someone has won vertically.
8. As a user, I can expect that the system will notify me and my opponent if someone has won diagonally.
9. As a user, I want to be notified by the system if there is a draw.
10. As a user, if I choose a position where a marker already has been placed, the system will tell me that I cannot place a marker there.

11. As a user, if I choose a position that is out of the bounds of the board, the system will tell me that it is not a valid position.
12. As a user, I want to be able to view both mine and my opponents placed markers after every turn.
13. As a user, I expect the top of the board to be the index 0, 0.
14. As a user, I want to be asked after the game has ended if I want to play again.
15. As a user, if I choose to play again, then the program should start over from the beginning and clear the game board.

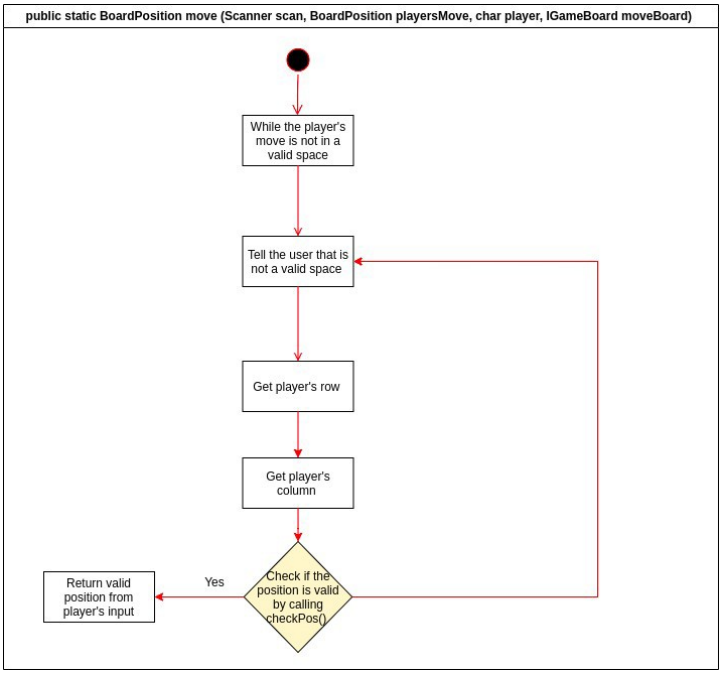
Non-Functional Requirements

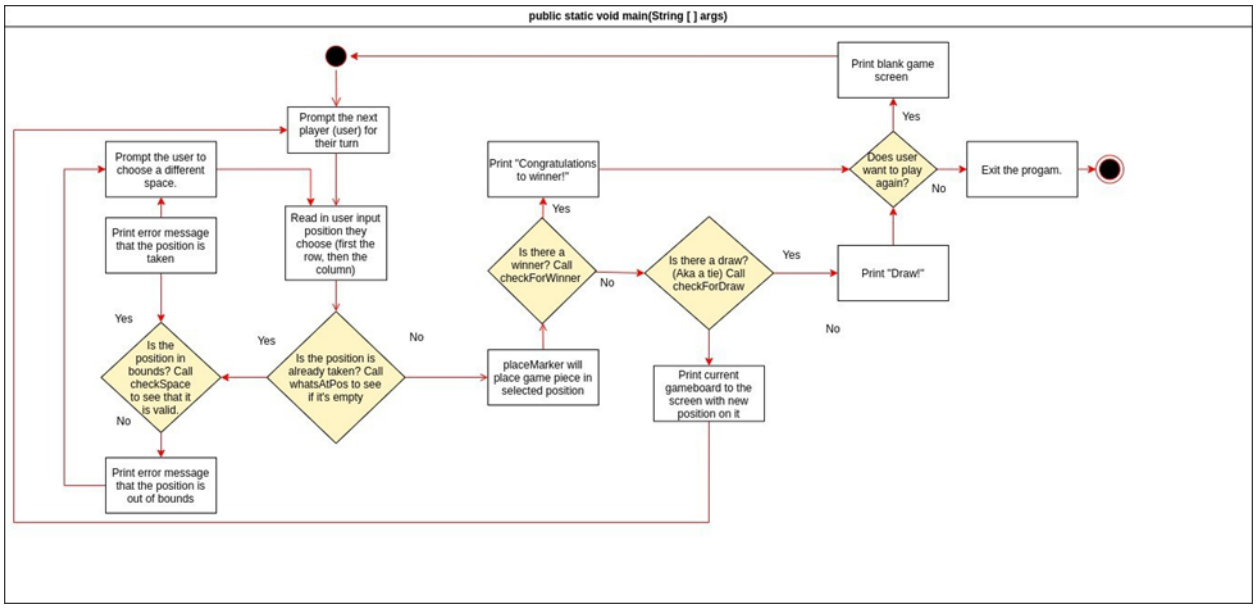
1. The system must be coded in Java programming language.
2. The system must be able to run on Unix/Linux.
3. Program must be able to compile and run quickly and efficiently.
4. The system must be written in IntelliJ IDE for debugging purposes in the future.
5. The top left corner of the game board must be 0, 0.
6. The game board is currently 8x8.
7. Currently, player X goes first and then player O.

Design

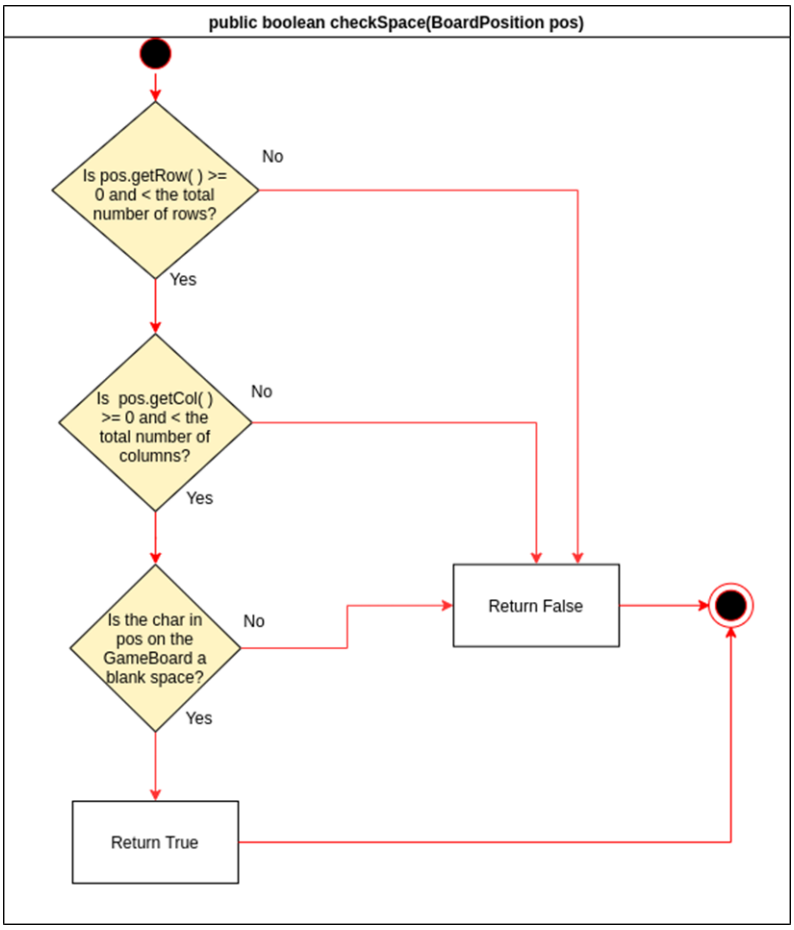
Activity Diagrams

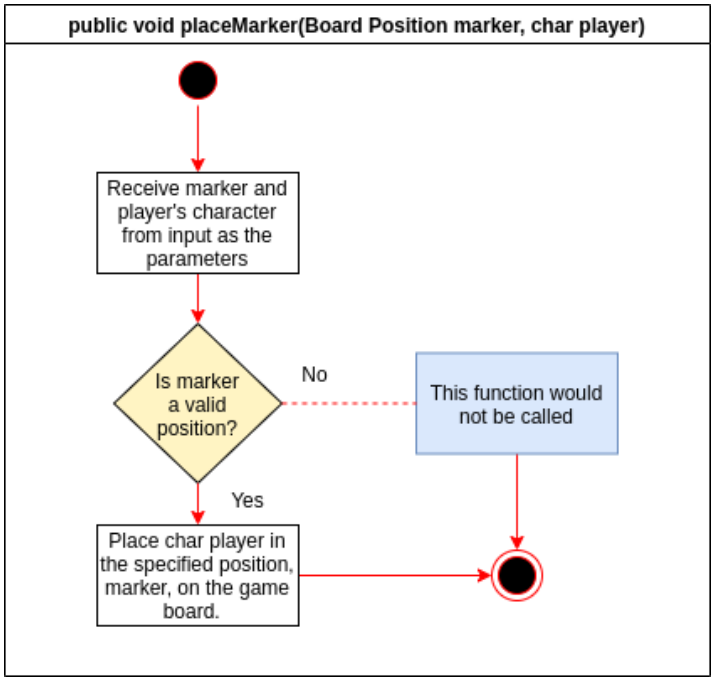
GameScreen.java

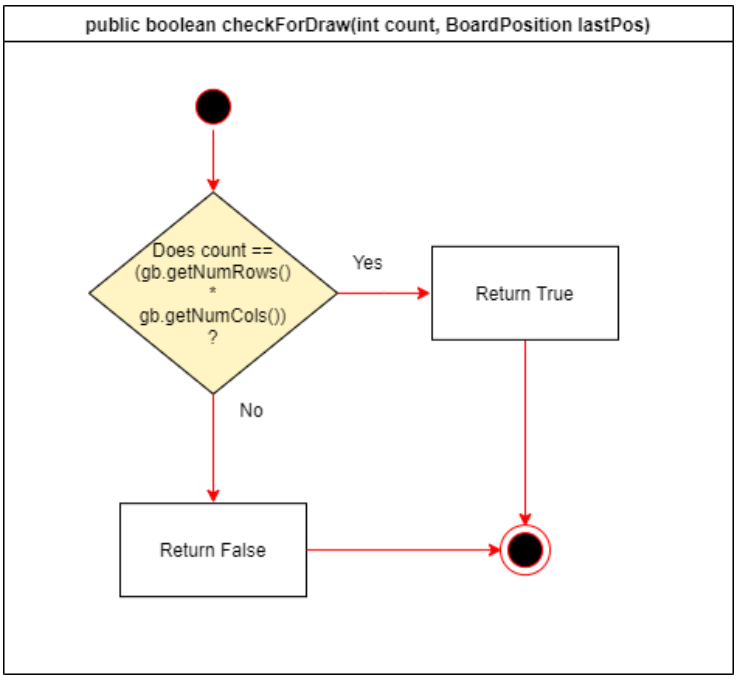


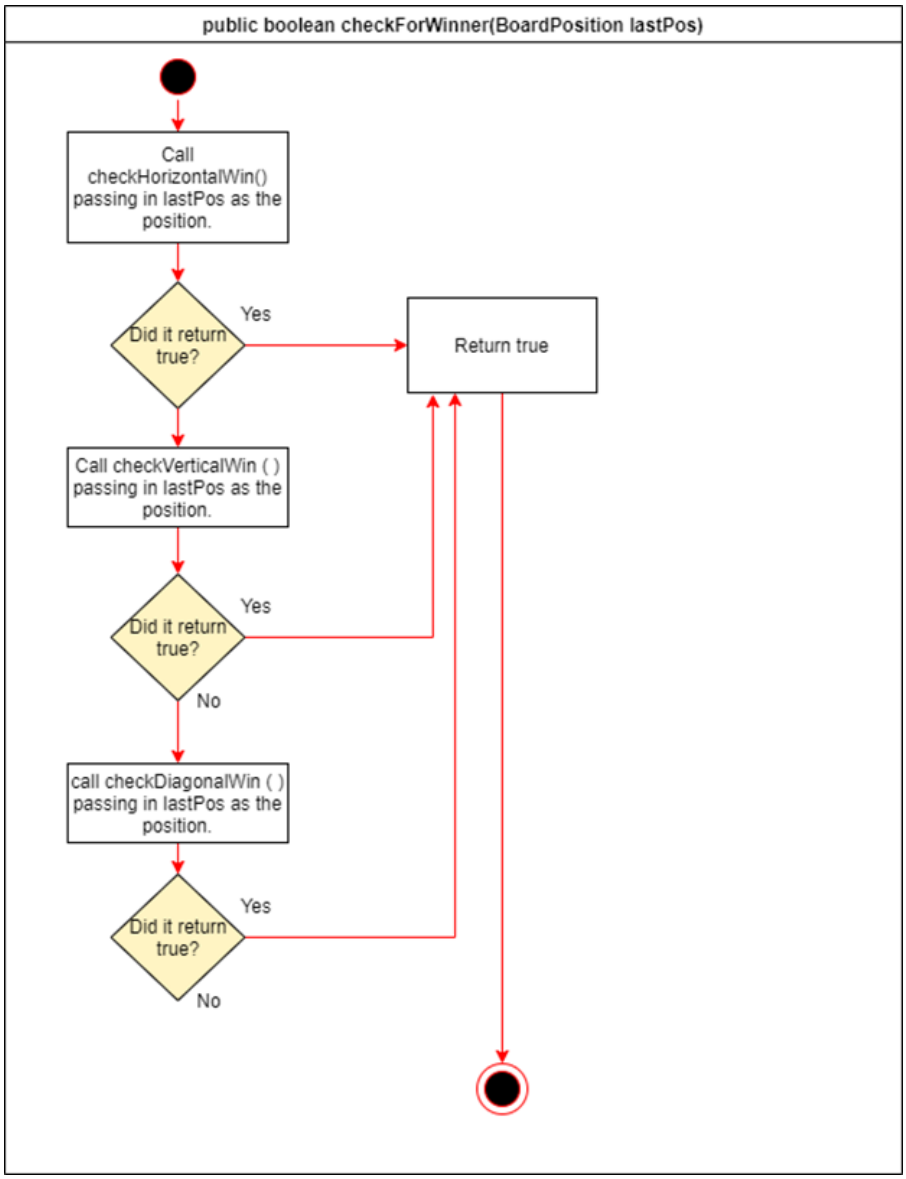


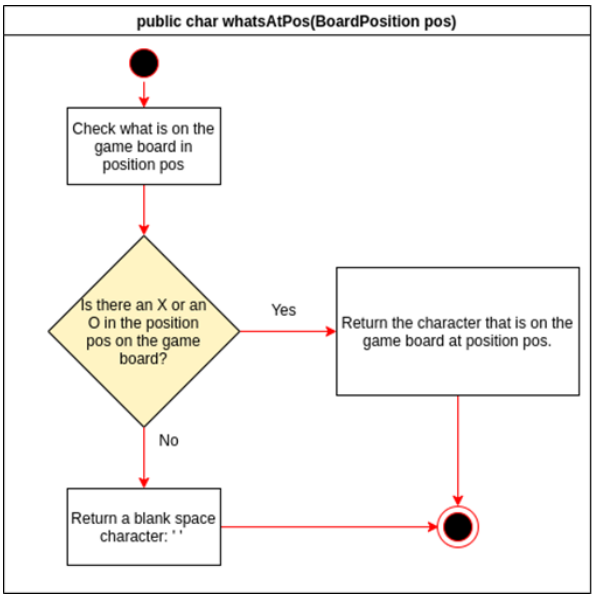
GameBoard.java

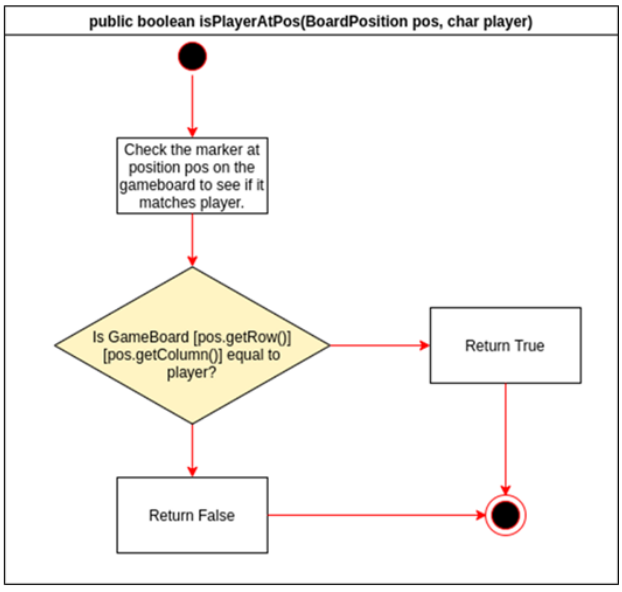


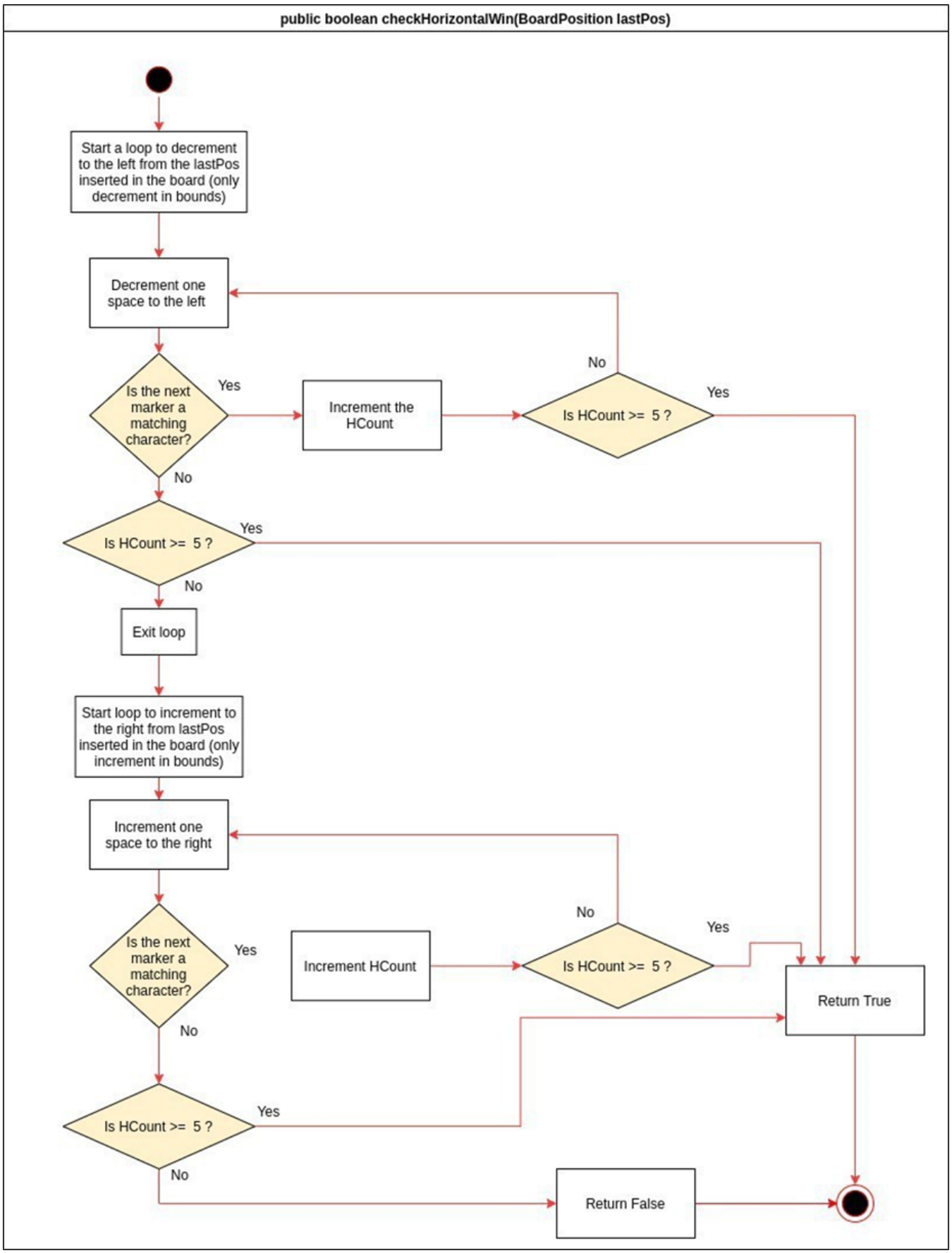


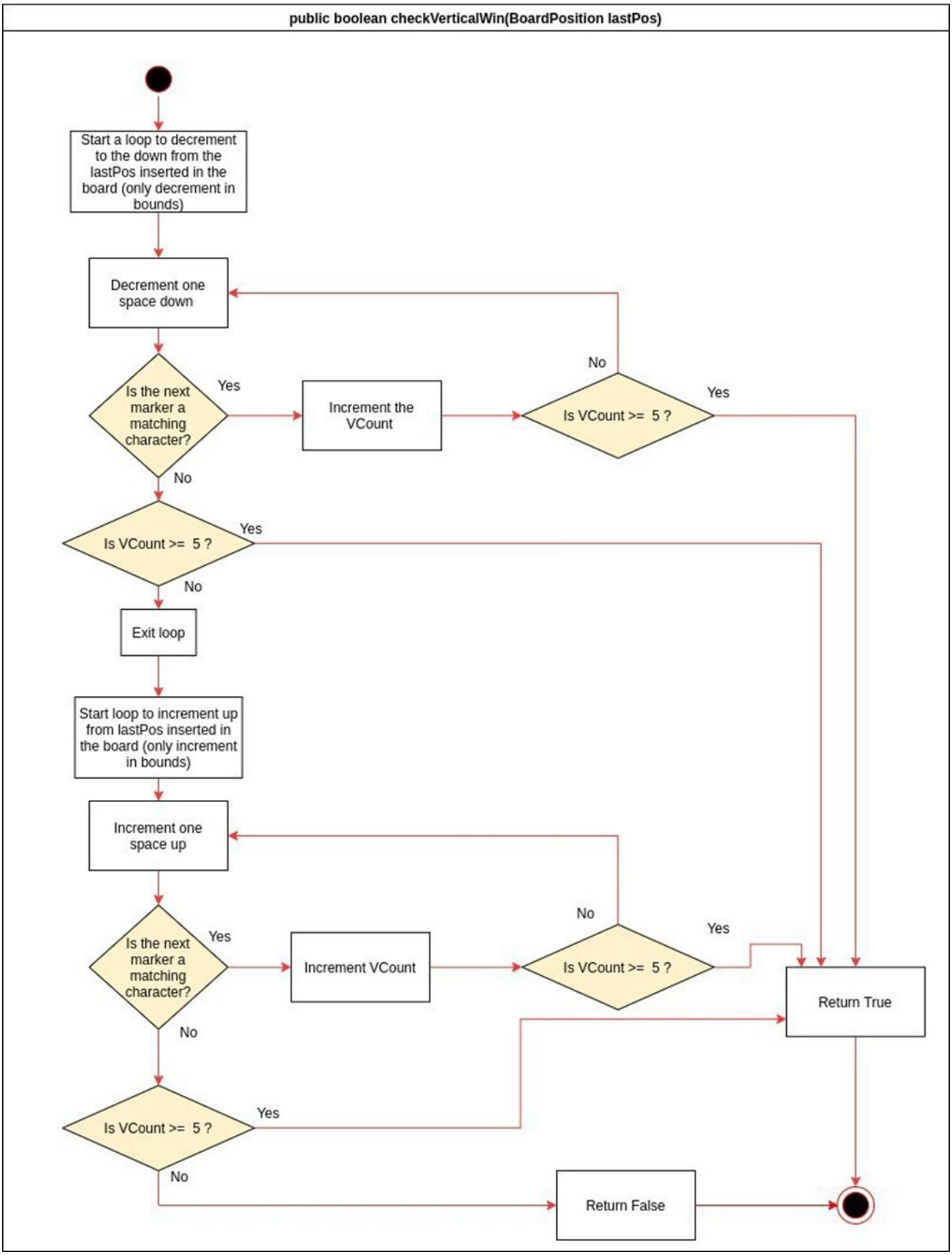


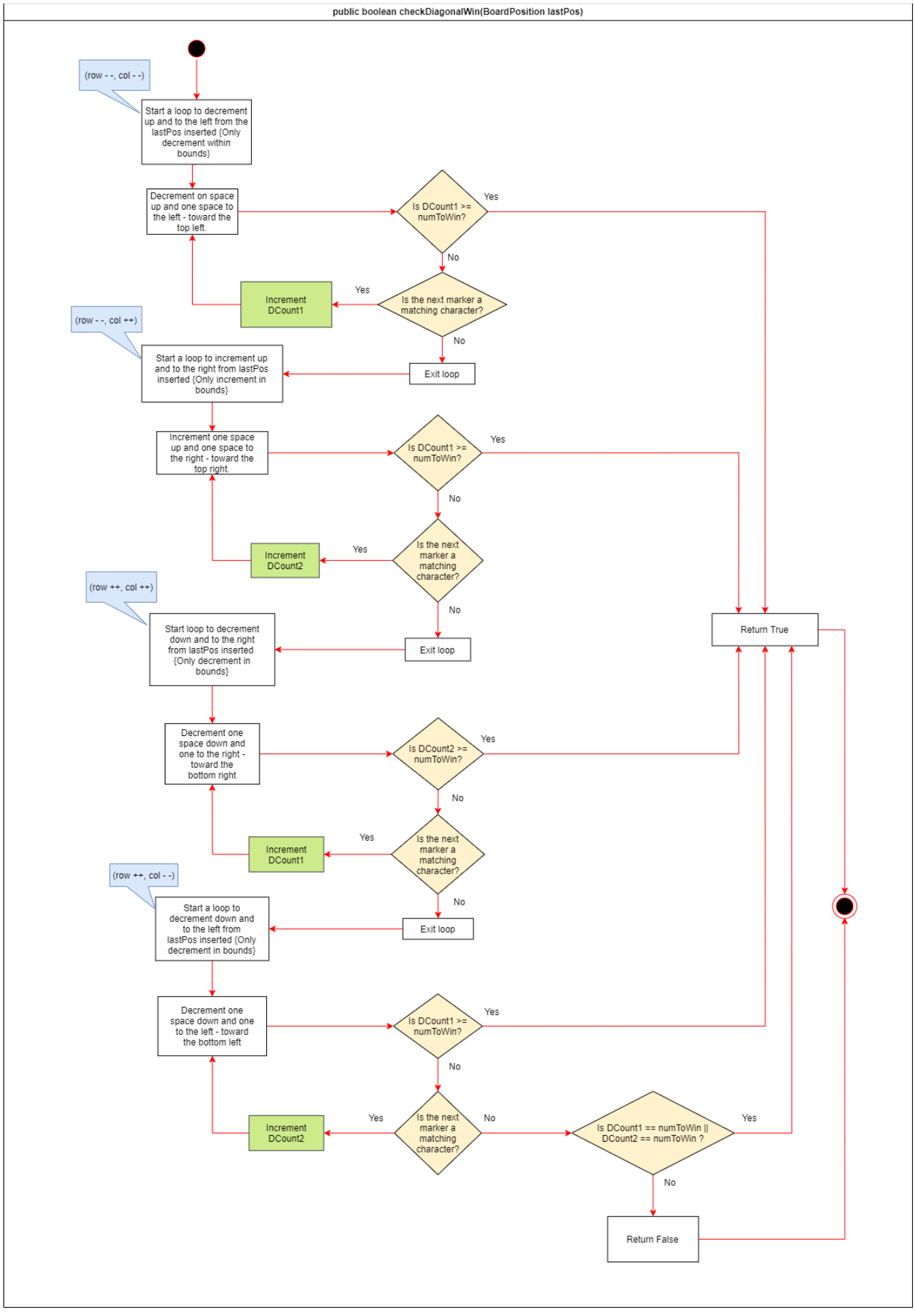

public boolean checkForWinner(BoardPosition lastPos)
Call checkHorizontalWin() passing in lastPos as the position.
Did it return true?
Yes
Return true
Call checkVerticalWin ( ) passing in lastPos as the position.
Did it return true?
Yes
No
call checkDiagonalWin ( ) passing in lastPos as the position.
Did it return true?
Yes
No

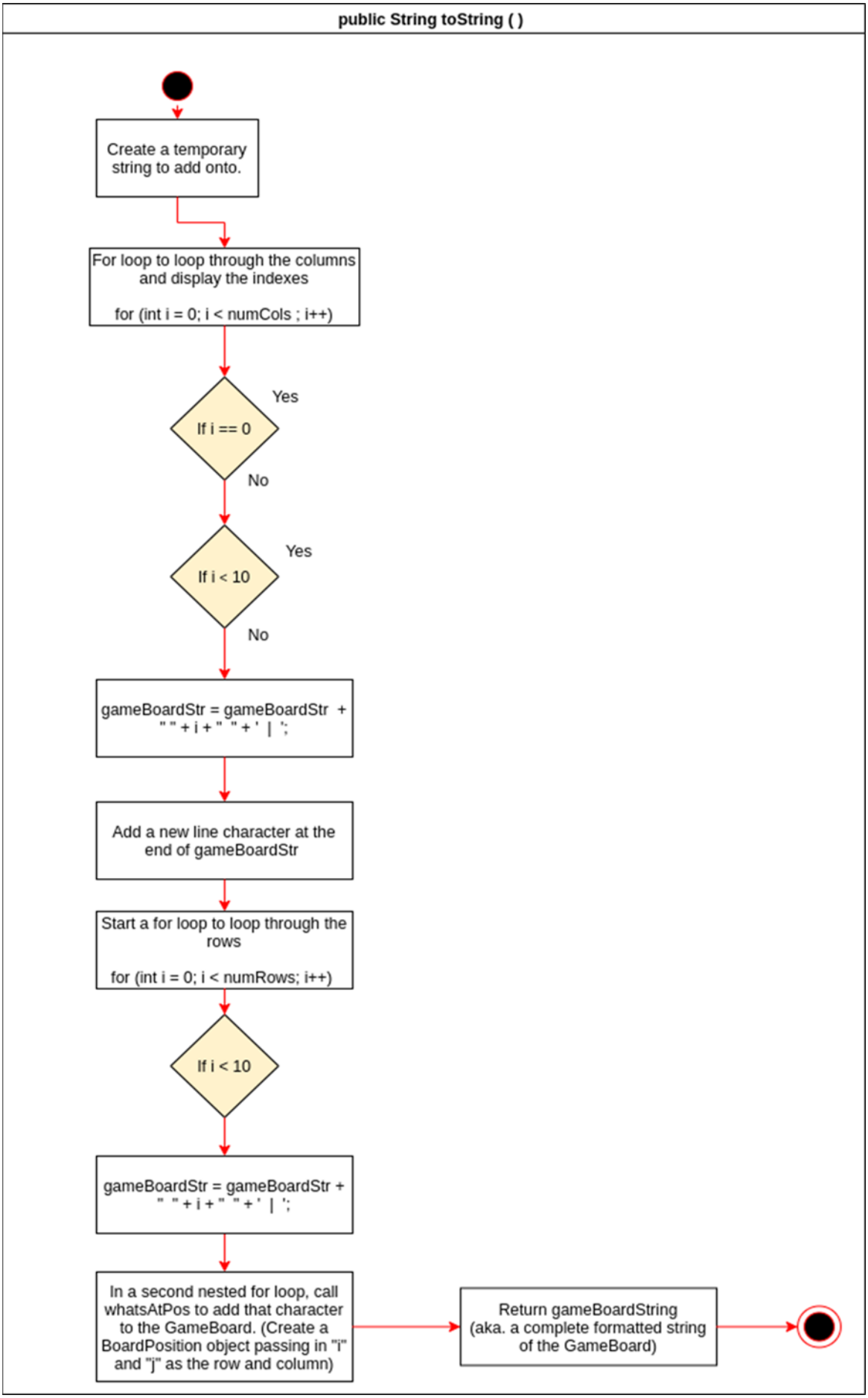

public char whatsAtPos(BoardPosition pos)
Check what is on the game board in position pos
Is there an X or an O in the position pos on the game board?
Yes
Return the character that is on the game board at position pos.
No
Return a blank space character: ' '


public boolean isPlayerAtPos(BoardPosition pos, char player)
Check the marker at position pos on the gameboard to see if it matches player.
Is GameBoard [pos.getRow()] [pos.getColumn()] equal to player?
Return True
Return False

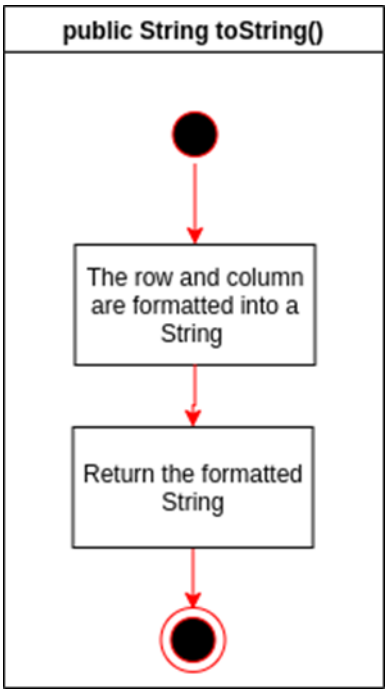

public boolean checkHorizontalWin(BoardPosition lastPos)
Start a loop to decrement to the left from the lastPos inserted in the board (only decrement in bounds)
Decrement one space to the left
Is the next marker a matching character?
Yes
Increment the HCount
Is HCount >= 5 ?
No
Yes
No
Is HCount >= 5 ?
Yes
No
Exit loop
Start loop to increment to the right from lastPos inserted in the board (only increment in bounds)
Increment one space to the right
Is the next marker a matching character?
Yes
Increment HCount
Is HCount >= 5 ?
No
Yes
No
Is HCount >= 5 ?
Yes
No
Return True
Return False

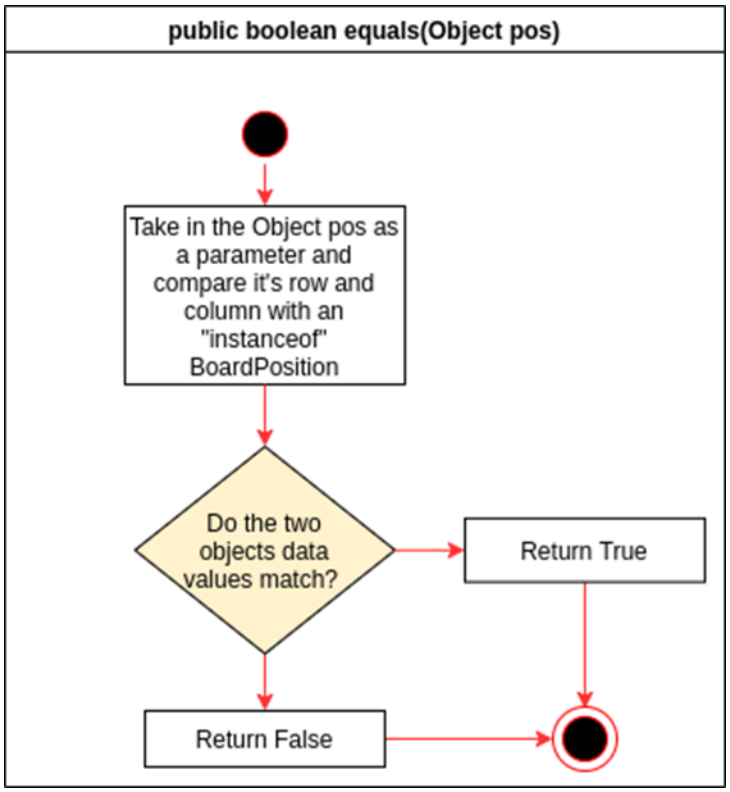

public boolean checkVerticalWin(BoardPosition lastPos)
Start a loop to decrement to the down from the lastPos inserted in the board (only decrement in bounds)
Decrement one space down
Is the next marker a matching character?
Yes
Increment the VCount
Is VCount >= 5 ?
Yes
No
No
Is VCount >= 5 ?
Yes
No
Exit loop
Start loop to increment up from lastPos inserted in the board (only increment in bounds)
Increment one space up
Is the next marker a matching character?
Yes
Increment VCount
Is VCount >= 5 ?
Yes
No
Return True
No
Is VCount >= 5 ?
Yes
No
Return False

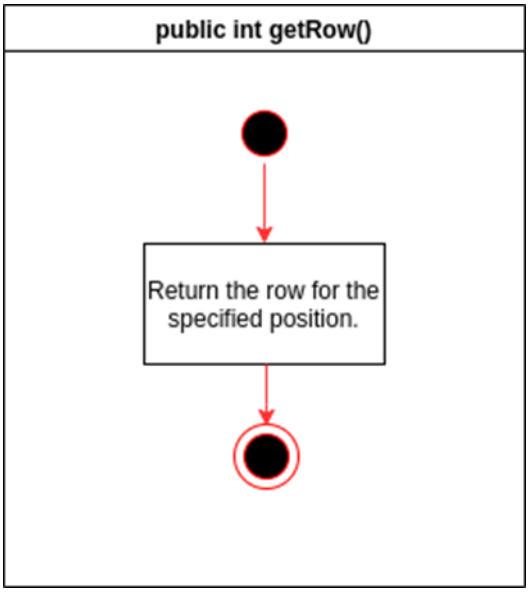

public boolean checkDiagonalWin(BoardPosition lastPos)
(row - -, col - -)
Start a loop to decrement up and to the left from the lastPos inserted {Only decrement within bounds}
Decrement on space up and one space to the left - toward the top left.
Is DCount1 >= numToWin?
Yes
No
Is the next marker a matching character?
Yes
Increment DCount1
No
Exit loop
(row - -, col ++)
Start a loop to increment up and to the right from lastPos inserted {Only increment in bounds}
Increment one space up and one space to the right - toward the top right.
Is DCount1 >= numToWin?
Yes
No
Is the next marker a matching character?
Yes
Increment DCount2
No
Exit loop
(row ++, col ++)
Start loop to decrement down and to the right from lastPos inserted {Only decrement in bounds}
Decrement one space down and one to the right - toward the bottom right
Is DCount2 >= numToWin?
Yes
No
Is the next marker a matching character?
Yes
Increment DCount1
No
Exit loop
(row ++, col - -)
Start a loop to decrement down and to the left from lastPos inserted {Only decrement in bounds}
Decrement one space down and one to the left - toward the bottom left
Is DCount1 >= numToWin?
Yes
No
Is the next marker a matching character?
Yes
Increment DCount2
No
Is DCount1 == numToWin || DCount2 == numToWin ?
Yes
No
Return False
Return True

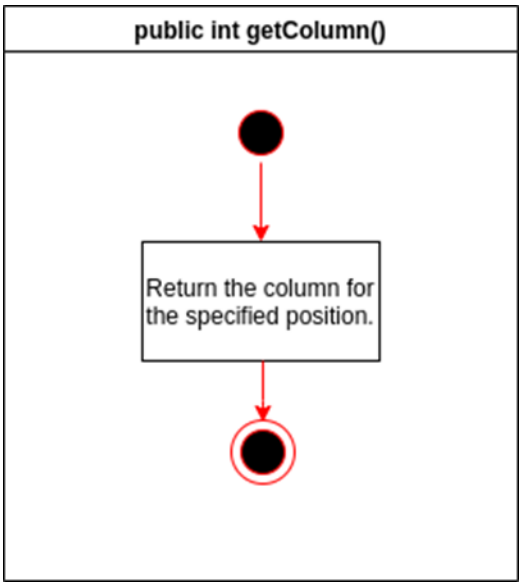

public String toString ( )
Create a temporary string to add onto.
For loop to loop through the columns and display the indexes
for (int i = 0; i < numCols ; i++)
If i == 0
Yes
No
If i < 10
Yes
No
gameBoardStr = gameBoardStr + " " + i + " " + ' | ';
Add a new line character at the end of gameBoardStr
Start a for loop to loop through the rows
for (int i = 0; i < numRows; i++)
If i < 10
gameBoardStr = gameBoardStr + " " + i + " " + ' | ';
In a second nested for loop, call whatsAtPos to add that character to the GameBoard. (Create a BoardPosition object passing in "i" and "j" as the row and column)
Return gameBoardString (aka. a complete formatted string of the GameBoard)

BoardPosition.java

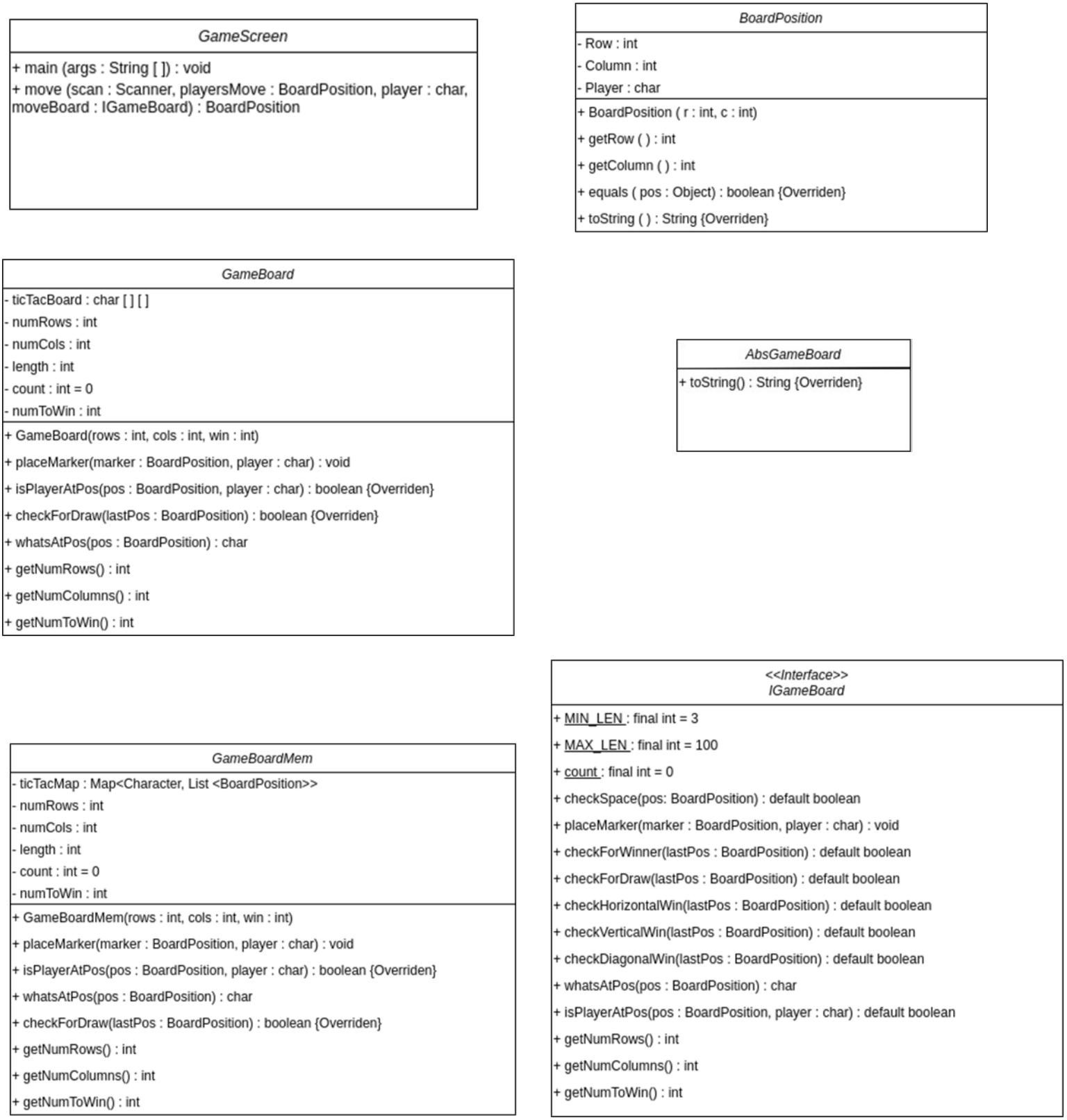


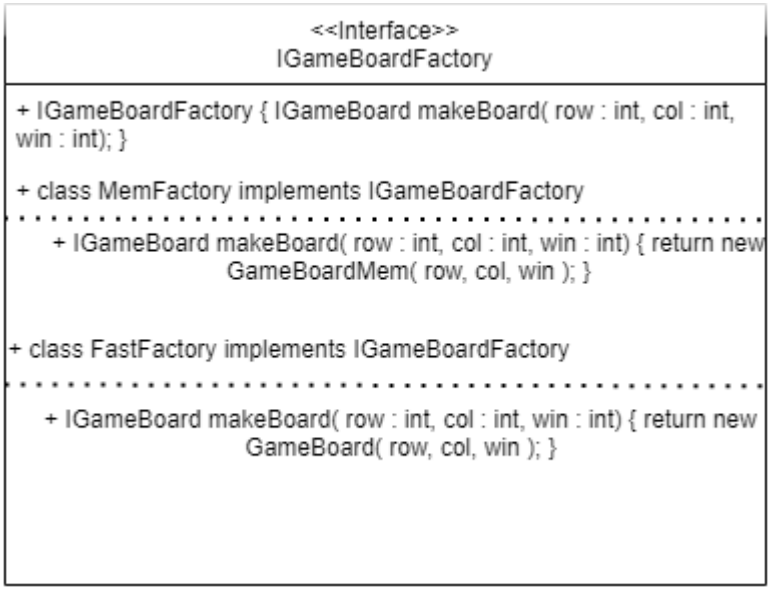






UML Class Diagrams





Testing

`public void GameBoard()`

Input:	**Output:**	**Reason:**
State: (number to win = 3) 0 1 2 0 1 2 IGameBoard gb = GameBoardFactory(3, 3, 3)	GameBoard constructor getNumRows() = 3 getNumCols() = 3 getNumToWin() = 3 State: unchanged	This test case is unique and distinct because it tests that the constructor successfully creates a game board using the minimum values for row, column, and number in a row to win. **Function Name:** `TestGameBoard_Minimums`

`public void GameBoard()`

Input:	**Output:**	**Reason:**
State: (number to win = 25) 0 1 2 0 1 2 IGameBoard gb = GameBoardFactory(100, 100, 25)	GameBoard constructor getNumRows() = 100 getNumCols() = 100 getNumToWin() = 25 State: unchanged	This test case is unique and distinct because it tests that the constructor successfully creates a game board using the maximum values for row, column, and number in a row to win. **Function Name:** `TestGameBoard_Maximums`

`public void GameBoard()`

Input:	**Output:**	**Reason:**
State: (number to win = 3) | | 0 | 1 | 2 | 3 | 4 | 5 | | 0 | | | | | | | | 1 | | | | | | | | 2 | | | | | | | | 3 | | | | | | | IGameBoard gb = GameBoardFactory(4, 6, 3)	GameBoard constructor getNumRows() = 4 getNumCols() = 6 getNumToWin() = 3 State: unchanged	This test case is unique and distinct because it tests that the constructor successfully creates a game board using different values for row and column. **Function Name:** `TestGameBoard_Diff_RowsAndCols`

`public void placeMarker(BoardPosition testPos, char player)`

Input:	**Output:**	**Reason:**
State: (number to win = 3) | | 0 | 1 | 2 | | 0 | | | | | 1 | | | | | 2 | | | | IGameBoard gb = GameBoardFactory(3, 3, 3) player = 'X' testPos = new BoardPosition (2, 2) gb.placeMarker (testPos, player)	getNumRows() = 3 getNumCols() = 3 getNumToWin() = 3 whatsAtPos(testPos) = 'X' State: | | 0 | 1 | 2 | | 0 | | | | | 1 | | | | | 2 | | | X |	This test case is unique and distinct because it tests that the placeMarker function can successfully place a player's character in the bottom right corner on the game board (2, 2). **Function Name:** `Test_PlaceMarker_in_Bottom_Right_Corner()`

`public void placeMarker(BoardPosition testPos, char player)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (0, 2)
gb.placeMarker (testPos, player)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = 'X'

State:

	0	1	2
0			X
1			
2			

Reason:
This test case is unique and distinct because it tests that the placeMarker function can successfully place a player's character in top right corner on the game board (0, 2).

Function Name:
`Test_PlaceMarker_in_Top_Right_Corner()`

`public void placeMarker(BoardPosition testPos, char player)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (2, 0)
gb.placeMarker (testPos, player)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = 'X'

State:

	0	1	2
0			
1			
2	X		

Reason:
This test case is unique and distinct because it tests that the placeMarker function can successfully place a player's character in the bottom left corner on the game board (2, 0).

Function Name:
`Test_PlaceMarker_in_Bottom_Left_Corner()`

`public void placeMarker(BoardPosition testPos, char player)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (0, 0)
gb.placeMarker (testPos, player)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = 'X'

State:

	0	1	2
0	X		
1			
2			

Reason:
This test case is unique and distinct because it tests that the placeMarker function can successfully place a player's character in top left corner on the game board (0, 0).

Function Name:
`Test_PlaceMarker_in_Top_Left_Corner()`

`public void placeMarker(BoardPosition testPos, char player)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (1, 1)
gb.placeMarker (testPos, player)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
use whatsAtPos(testPos) for comparing the results of a cell in the board after a marker is placed.

State:

	0	1	2
0	X	O	X
1	O	X	O
2	X	O	X

Reason:
This test case is unique and distinct because it tests that the placeMarker function can successfully place a player's character in the last available position on the game board (1, 1).

Function Name:
`Test_PlaceMarker_Last_Marker_on_Board()`

`public char whatsAtPos(BoardPosition testPos)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (2, 2)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = ' '

State:

	0	1	2
0			
1			
2			

Reason:
This test case is unique and distinct because it tests that the whatsAtPos function can successfully determine and then return the character that is in the specified position.

Function Name:
`Test_WhatsAtPos_Blank_Board()`

`public char whatsAtPos(BoardPosition testPos)`

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (0, 0)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = 'X'

State:

	0	1	2
0	X		
1			
2			

Reason:
This test case is unique and distinct because it tests that the whatsAtPos function can successfully determine and then return the character that is in the specified position even if it is the only marker on the board.

Function Name:
`Test_WhatsAtPos_One_Marker_Placed()`

```
public char whatsAtPos(BoardPosition testPos)
```

Input:

State: (number to win = 3)

	0	1	2
0			
1			
2			

IGameBoard gb = GameBoardFactory(3, 3, 3)
player = 'X'

testPos = new BoardPosition (0, 0)

Output:

getNumRows() = 3
getNumCols() = 3
getNumToWin() = 3
whatsAtPos(testPos) = 'X'

State:

	0	1	2
0			
1			
2			X

Reason:
This test case is unique and distinct because it tests that the whatsAtPos function can successfully determine and then return the character that is in the specified position even if it is the only marker on the board.

Function Name:
`Test_WhatsAtPos_Bottom_Right_Corner()`