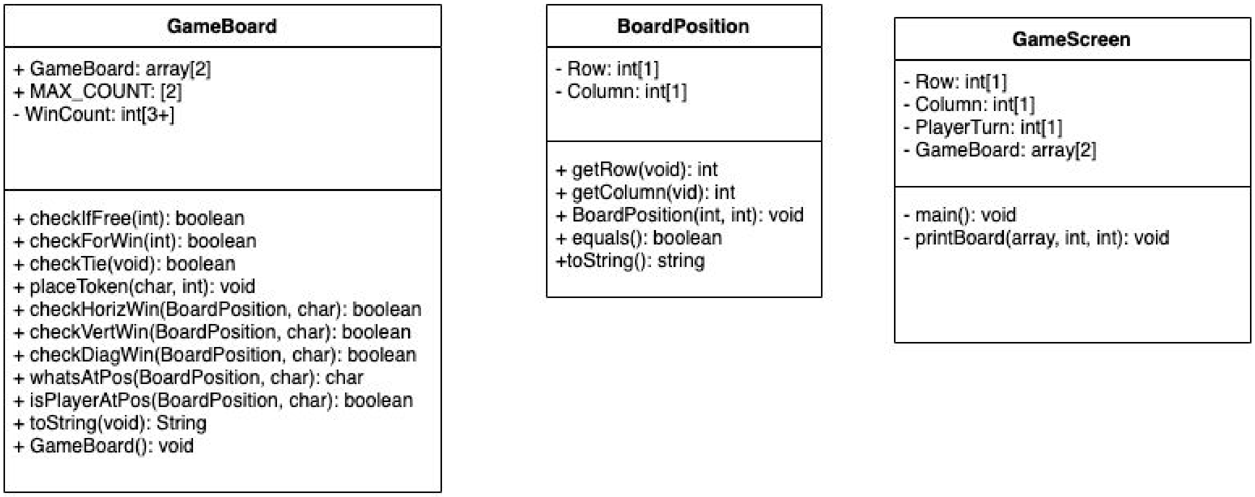
Annie Hayes ahayes5@g.clemson.edu

functional requirements:

1. As a player, I can only place a marker vertically in a column to simulate the connect 4 game.
2. As a user, I can only place a marker in 9 columns because there are only 9 columns
3. As a player, I cannot place a marker in a column that is already full to prevent overflow.
4. As a user, I can win if I have 5 of my markers in a row horizontally.
5. As a player, I can win if I have 5 of my markers in a row vertically.
6. As a player, I can win if I have 5 of my markers in a row diagonally.
7. As a strategist, I can have a tie in the game because all of the columns are full.
8. As the user, I can alternate between players so each player can have a turn.
9. As a player, I can see whose turn it is so I know who is supposed to pick a column
10. As a connect-4 pro, I can pick which column to place my marker so I know which spot I played.
11. As a player, I cannot pick a spot outside of the bounds of the board or I will get an error message.
12. As a player, I can see if I have won by looking if I have 4 in a row.
13. As a player, I can play again once the game has ended.
14. As a player, I can enter an integer value to say which column I have selected.
15. As a player, I cannot enter a value over 9 since there are only 9 columns.
16. As a player, X will start the game so it is consistent every game.
17. As a player, the board will keep track of all of the markers so I can see which positions are filled.

non-functional requirements

1. The system must be coded in Java
2. The system must run onUnix
3. Do not use magic numbers
4. use good comments
5. write contracts
6. make a program report
7. make UML class diagrams
8. make UML activity diagrams
9. write code for functions



GameBoard

+ GameBoard: array[2]
+ MAX_COUNT: [2]
- WinCount: int[3+]

+ checkIfFree(int): boolean
+ checkForWin(int): boolean
+ checkTie(void): boolean
+ placeToken(char, int): void
+ checkHorizWin(BoardPosition, char): boolean
+ checkVertWin(BoardPosition, char): boolean
+ checkDiagWin(BoardPosition, char): boolean
+ whatsAtPos(BoardPosition, char): char
+ isPlayerAtPos(BoardPosition, char): boolean
+ toString(void): String
+ GameBoard(): void

BoardPosition

- Row: int[1]
- Column: int[1]

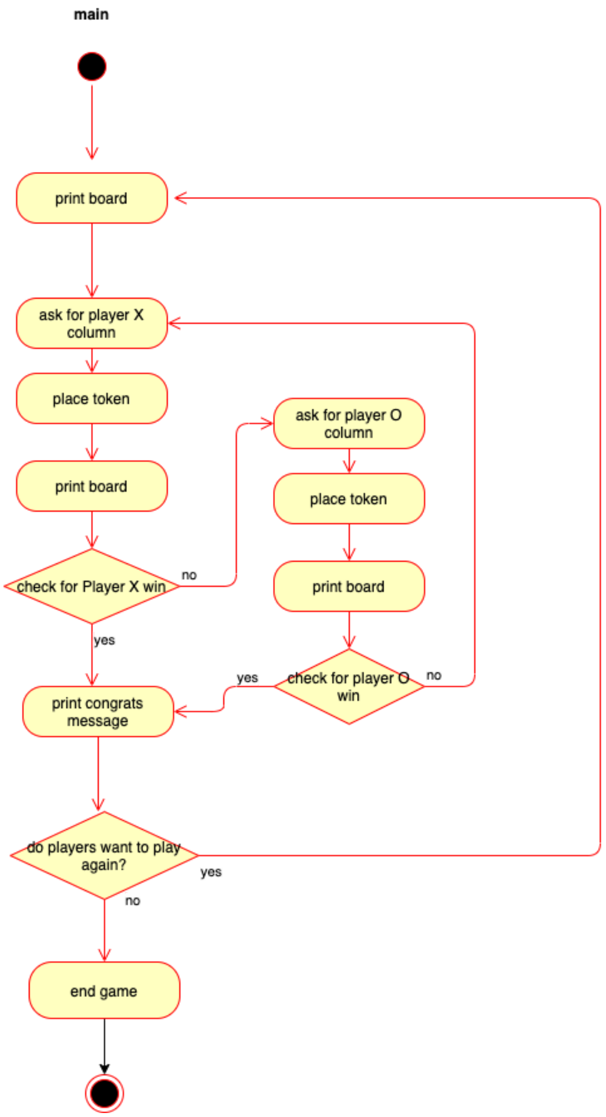
+ getRow(void): int
+ getColumn(vid): int
+ BoardPosition(int, int): void
+ equals(): boolean
+toString(): string

GameScreen

- Row: int[1]
- Column: int[1]
- PlayerTurn: int[1]
- GameBoard: array[2]

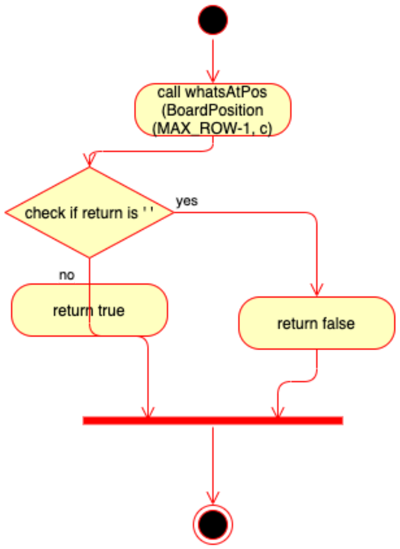
- main(): void
- printBoard(array, int, int): void

GameScreen.java

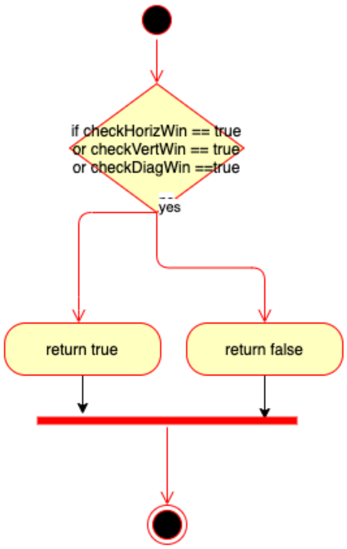


GameBoard.java

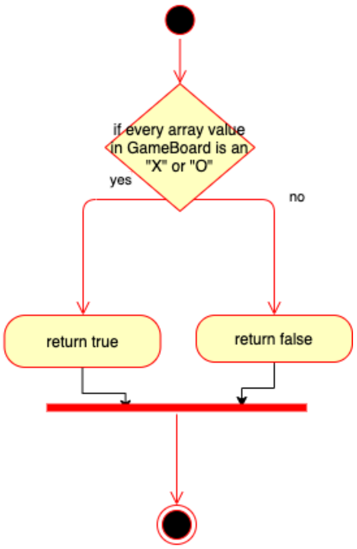
checkIfFree



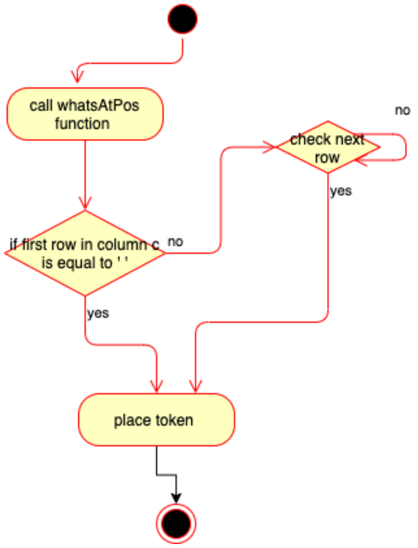
checkForWin


if checkHorizWin == true or checkVertWin == true or checkDiagWin ==true
yes
return true
return false

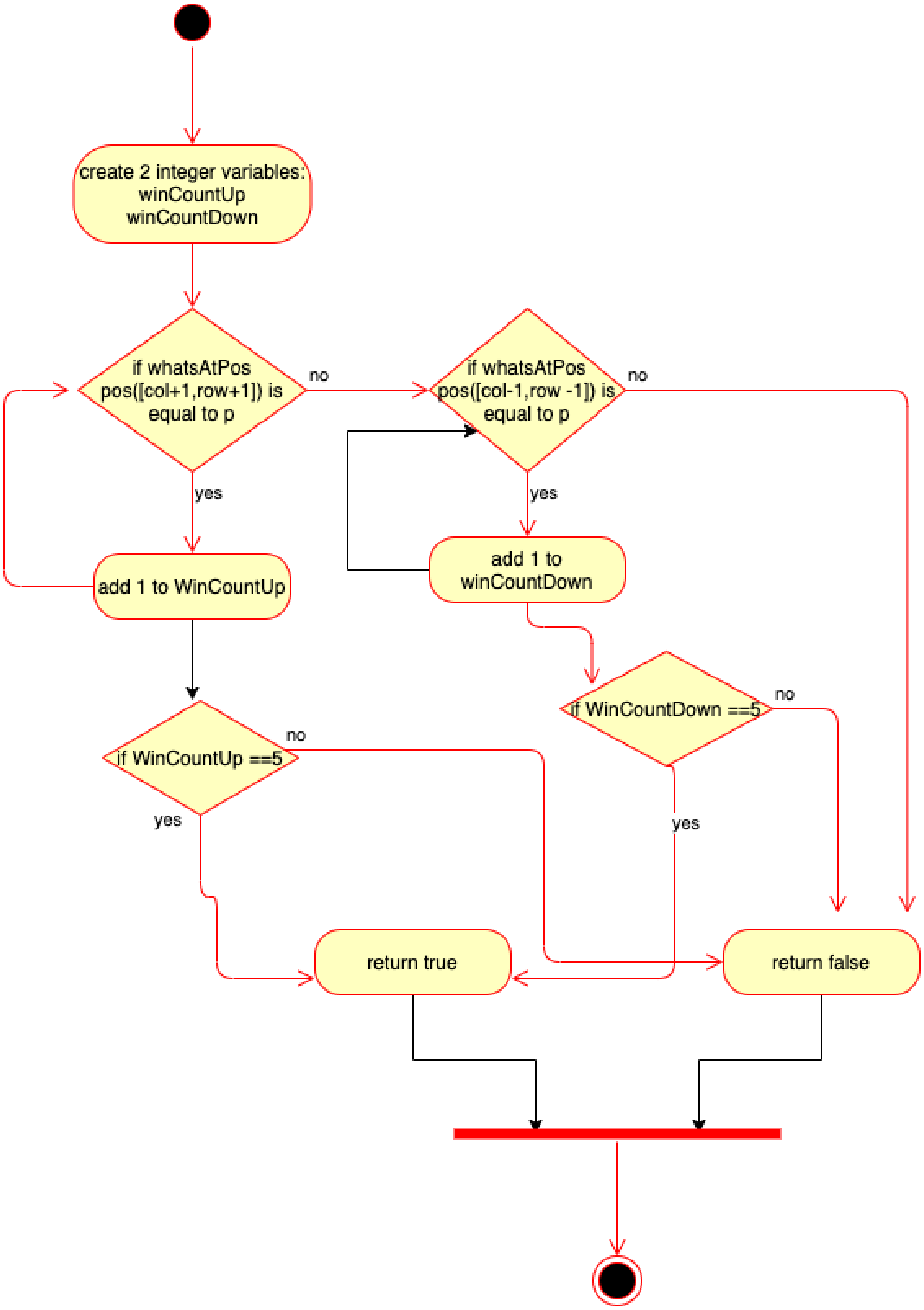
CheckTie



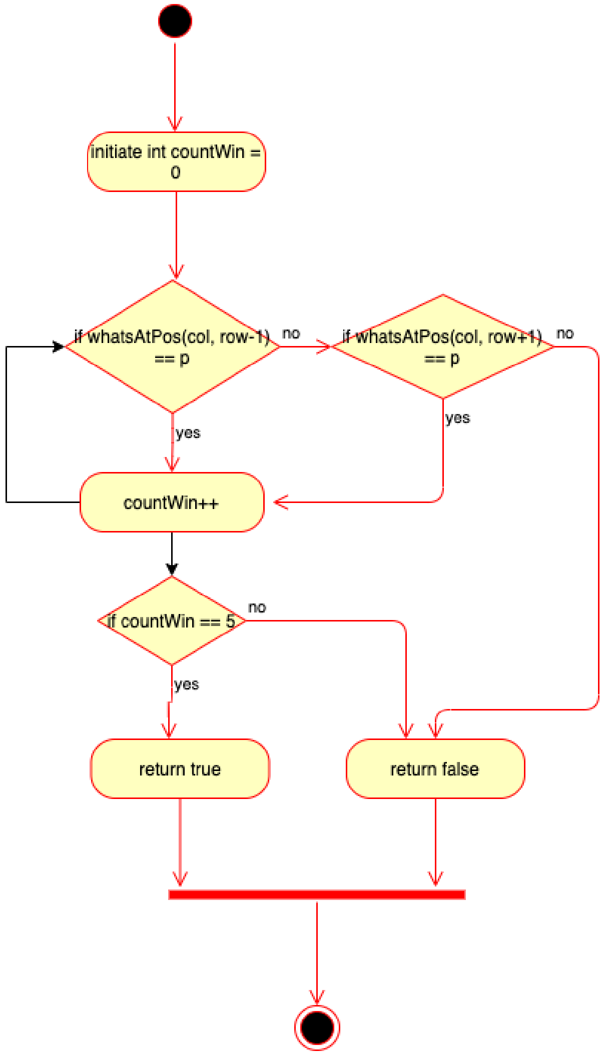
placeToken



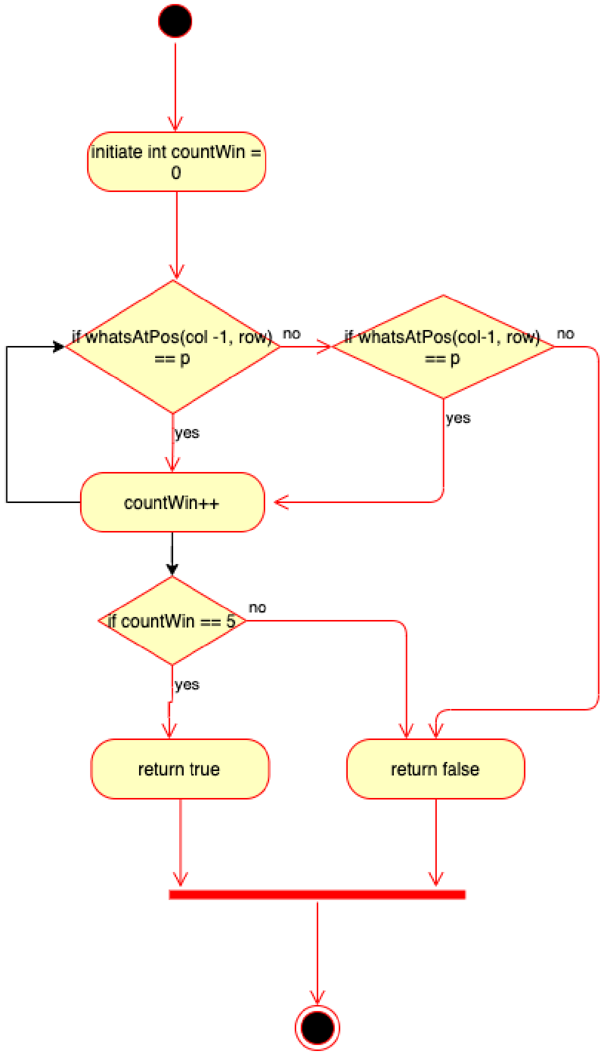
CheckDiagWin



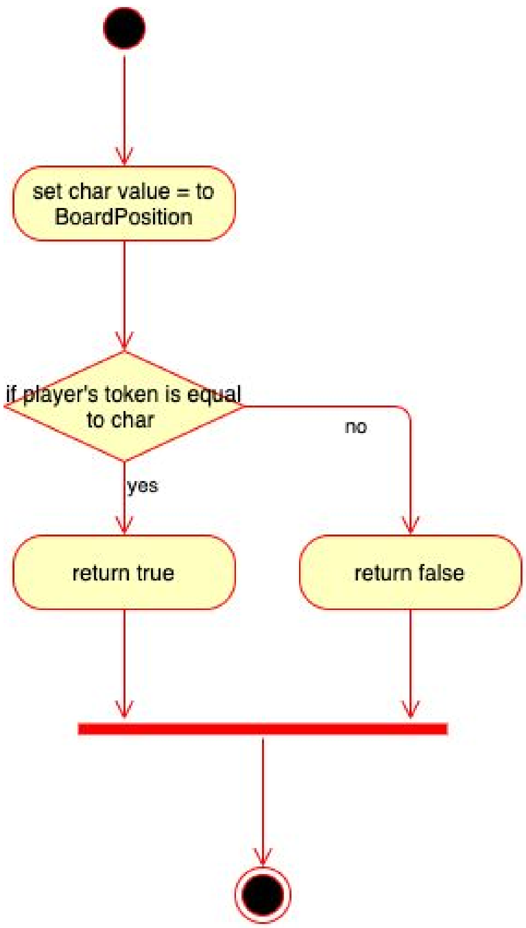
CheckVertWin


initiate int countWin = 0
if whatsAtPos(col, row-1) == p
no
if whatsAtPos(col, row+1) == p
no
yes
yes
countWin++
if countWin == 5
no
yes
return true
return false

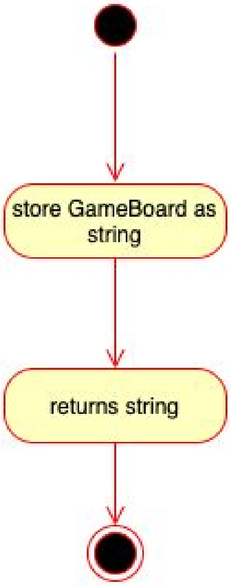
CheckHorizWin


initiate int countWin = 0
if whatsAtPos(col -1, row) == p
no
if whatsAtPos(col-1, row) == p
no
yes
yes
countWin++
if countWin == 5
no
yes
return true
return false

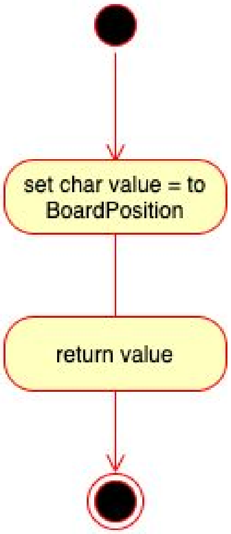
isPlayerAtPos


set char value = to BoardPosition
if player's token is equal to char
no
yes
return true
return false

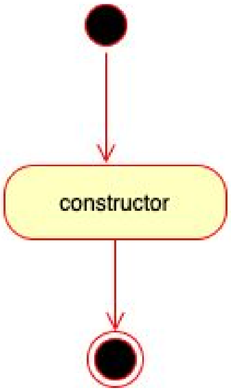
toString



whatsAtPos



GameBoard


constructor