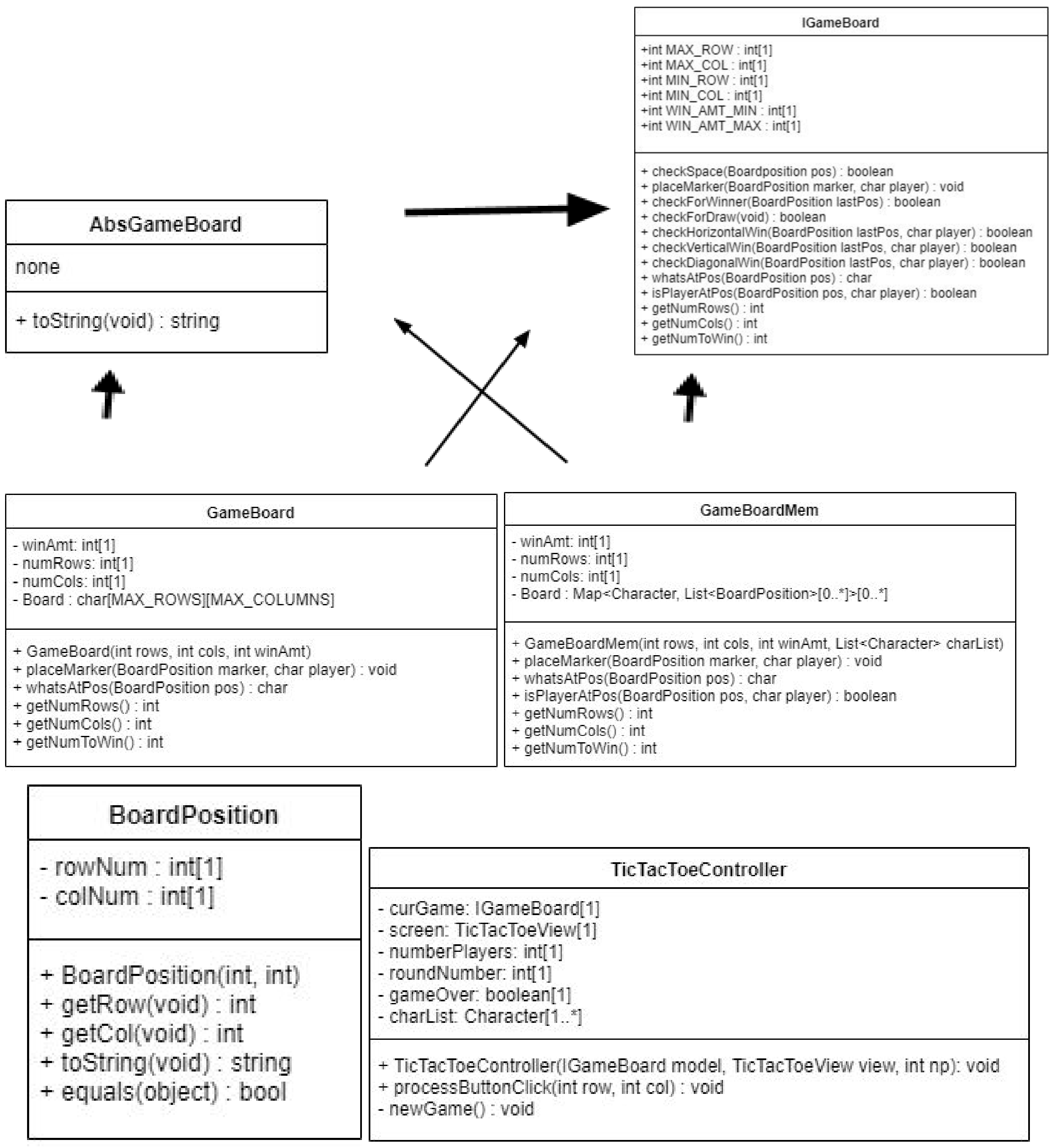
Requirements Analysis

User Stories

- Functional:
    - As a player, I need to be able to pick the number of rows and columns of the gameboard, so that I can customize the game.
    - As a player, I need to be able to pick the number of markers in a row to win, so that I can customize the game.
    - As a player, I can choose a new row and column if they aren't valid, so that I can play the game.
    - As a player, I can enter a win amount that is less than the number of rows or columns, so that it is possible to win.
    - As a player, I need to be able to pick a spot to put my marker, so I can play the game.
    - As a player, I need to know how to play the game, so that I can know how to win.
    - As a player, I need to know when I have won, so there can be a winner.
    - As a player, I need to know when it is my turn, so I can keep the game organized.
    - As a player, I need to have the option to play again, so I can have multiple rounds.
    - As a player, I need to know what the board looks like after I place a marker, so I don't have to imagine it.
    - As a player, I need to be sure there is no way to cheat, so the game can be played fair.
    - As a player, I can win by placing a certain amount in a row horizontally, so that I can win horizontally.
    - As a player, I can win by placing a certain amount in a row vertically, so that I can win vertically.
    - As a player, I can win by placing a certain amount in a row in both diagonals, so that I can win diagonally.
    - As a player, I can pick again if I pick an unavailable space, so that I can complete the game.
    - As a player, I can pick again if I pick a space that does not exist, so that I can complete the game.
    - As a player, I can play after my opponent does if he doesn't win, so that we can each have a turn.
    - As a player, I can end the game in a tie, so that there is another way to finish the game.
    - As a player, I can pick the number of players, so that we can play with between 2 and 10 players.
    - As a player, I can have my own marker, so that I can have a unique player.
    - As a player, I can choose to use the fast implementation or the memory efficient implementation, so I have the ability to save memory.
    - As a player, I can change all options when I start a new game, so that I can play again.

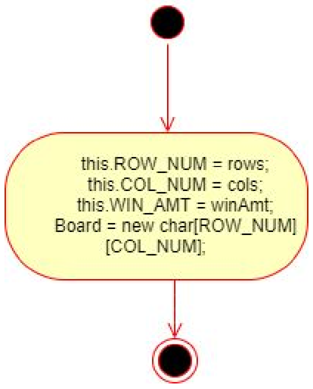
- Nonfunctional:
    - Written in Java.
    - Decently fast response unless memory implementation - efficient.
    - Able to be adapted.
    - 0,0 should be top left corner.

UML Class Diagrams

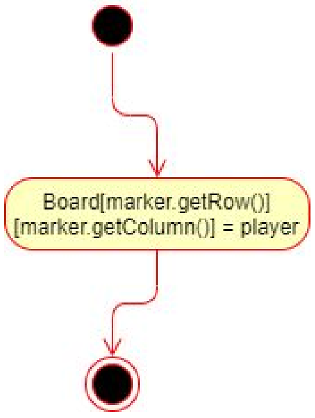

IGameBoard
+int MAX_ROW : int[1]
+int MAX_COL : int[1]
+int MIN_ROW : int[1]
+int MIN_COL : int[1]
+int WIN_AMT_MIN : int[1]
+int WIN_AMT_MAX : int[1]
+ checkSpace(Boardposition pos) : boolean
+ placeMarker(BoardPosition marker, char player) : void
+ checkForWinner(BoardPosition lastPos) : boolean
+ checkForDraw(void) : boolean
+ checkHorizontalWin(BoardPosition lastPos, char player) : boolean
+ checkVerticalWin(BoardPosition lastPos, char player) : boolean
+ checkDiagonalWin(BoardPosition lastPos, char player) : boolean
+ whatsAtPos(BoardPosition pos) : char
+ isPlayerAtPos(BoardPosition pos, char player) : boolean
+ getNumRows() : int
+ getNumCols() : int
+ getNumToWin() : int
AbsGameBoard
none
+ toString(void) : string
GameBoard
- winAmt: int[1]
- numRows: int[1]
- numCols: int[1]
- Board : char[MAX_ROWS][MAX_COLUMNS]
+ GameBoard(int rows, int cols, int winAmt)
+ placeMarker(BoardPosition marker, char player) : void
+ whatsAtPos(BoardPosition pos) : char
+ getNumRows() : int
+ getNumCols() : int
+ getNumToWin() : int
GameBoardMem
- winAmt: int[1]
- numRows: int[1]
- numCols: int[1]
- Board : Map<Character, List<BoardPosition>[0..*]>[0..*]
+ GameBoardMem(int rows, int cols, int winAmt, List<Character> charList)
+ placeMarker(BoardPosition marker, char player) : void
+ whatsAtPos(BoardPosition pos) : char
+ isPlayerAtPos(BoardPosition pos, char player) : boolean
+ getNumRows() : int
+ getNumCols() : int
+ getNumToWin() : int
BoardPosition
- rowNum : int[1]
- colNum : int[1]
+ BoardPosition(int, int)
+ getRow(void) : int
+ getCol(void) : int
+ toString(void) : string
+ equals(object) : bool
TicTacToeController
- curGame: IGameBoard[1]
- screen: TicTacToeView[1]
- numberPlayers: int[1]
- roundNumber: int[1]
- gameOver: boolean[1]
- charList: Character[1..*]
+ TicTacToeController(IGameBoard model, TicTacToeView view, int np): void
+ processButtonClick(int row, int col) : void
- newGame() : void

UML Activity Diagrams

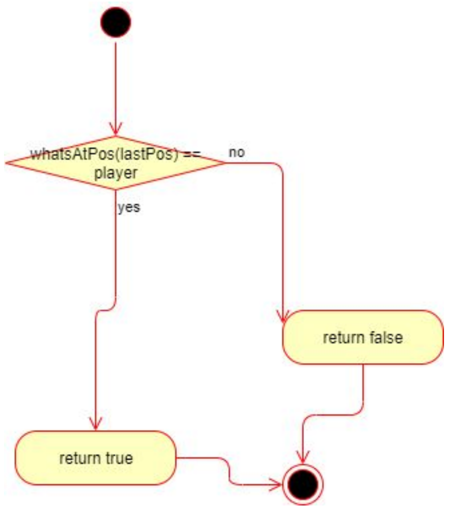
public GameBoard(int rows, int cols, int winAmt)



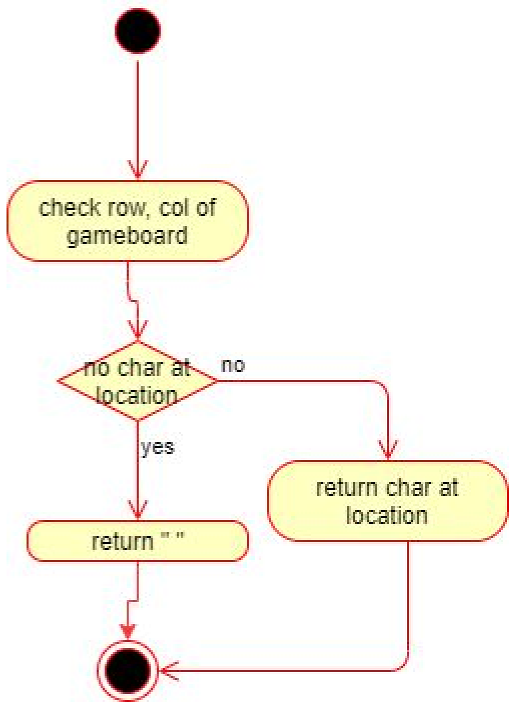
public void placeMarker(BoardPosition marker, char player)



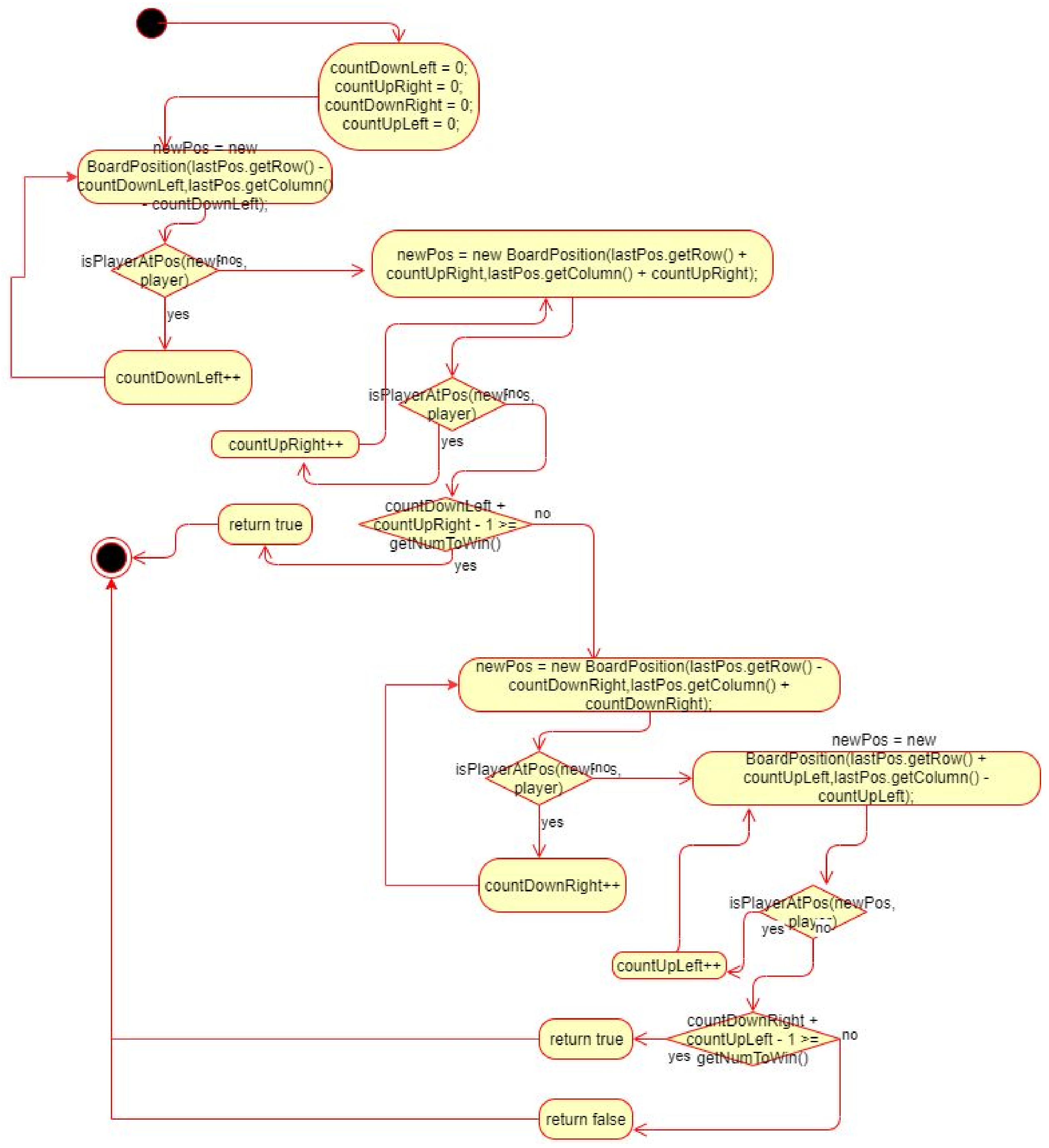
public boolean isPlayerAtPos(BoardPosition pos, char player)



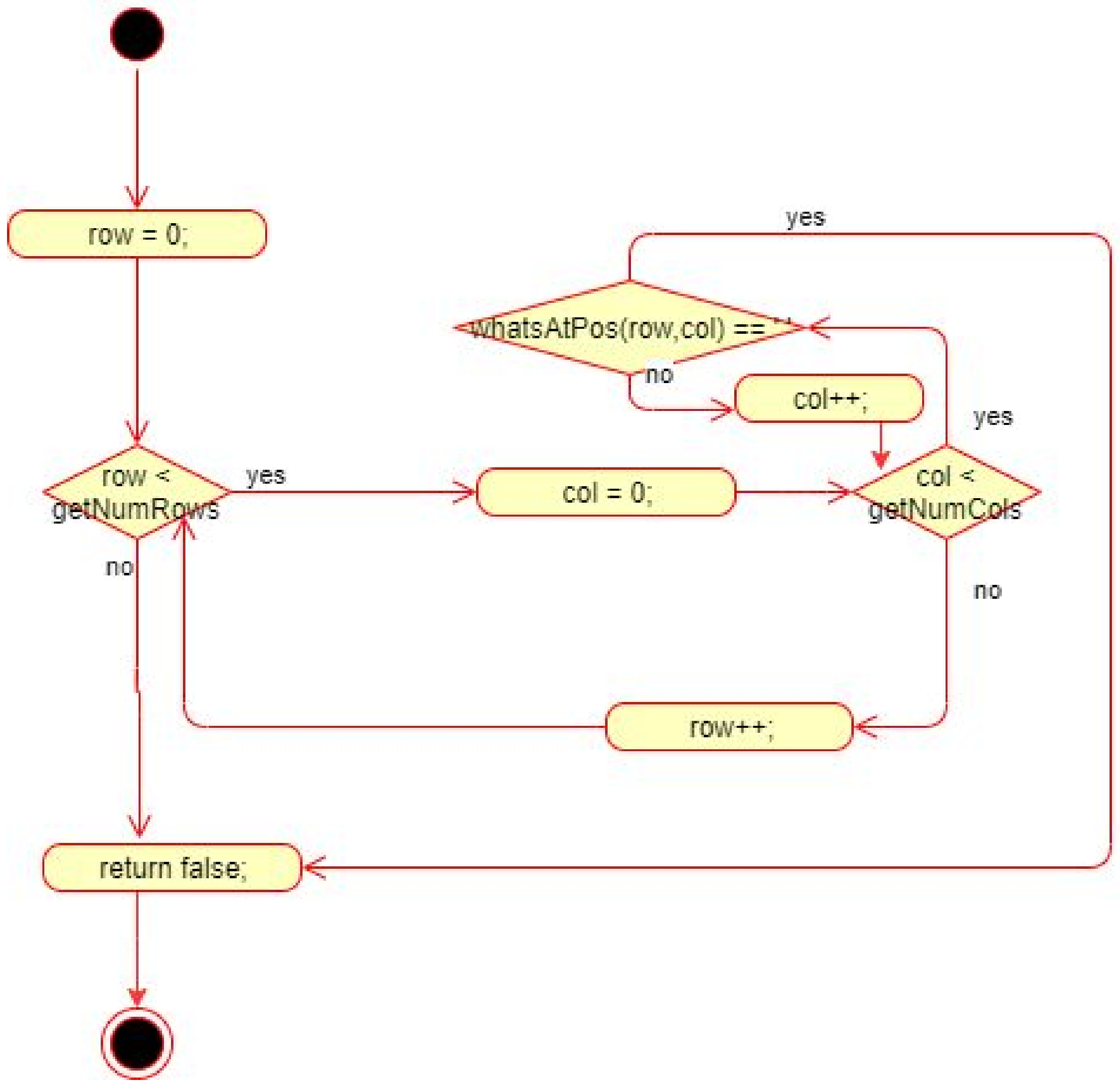
public char whatsAtPos(BoardPosition pos)



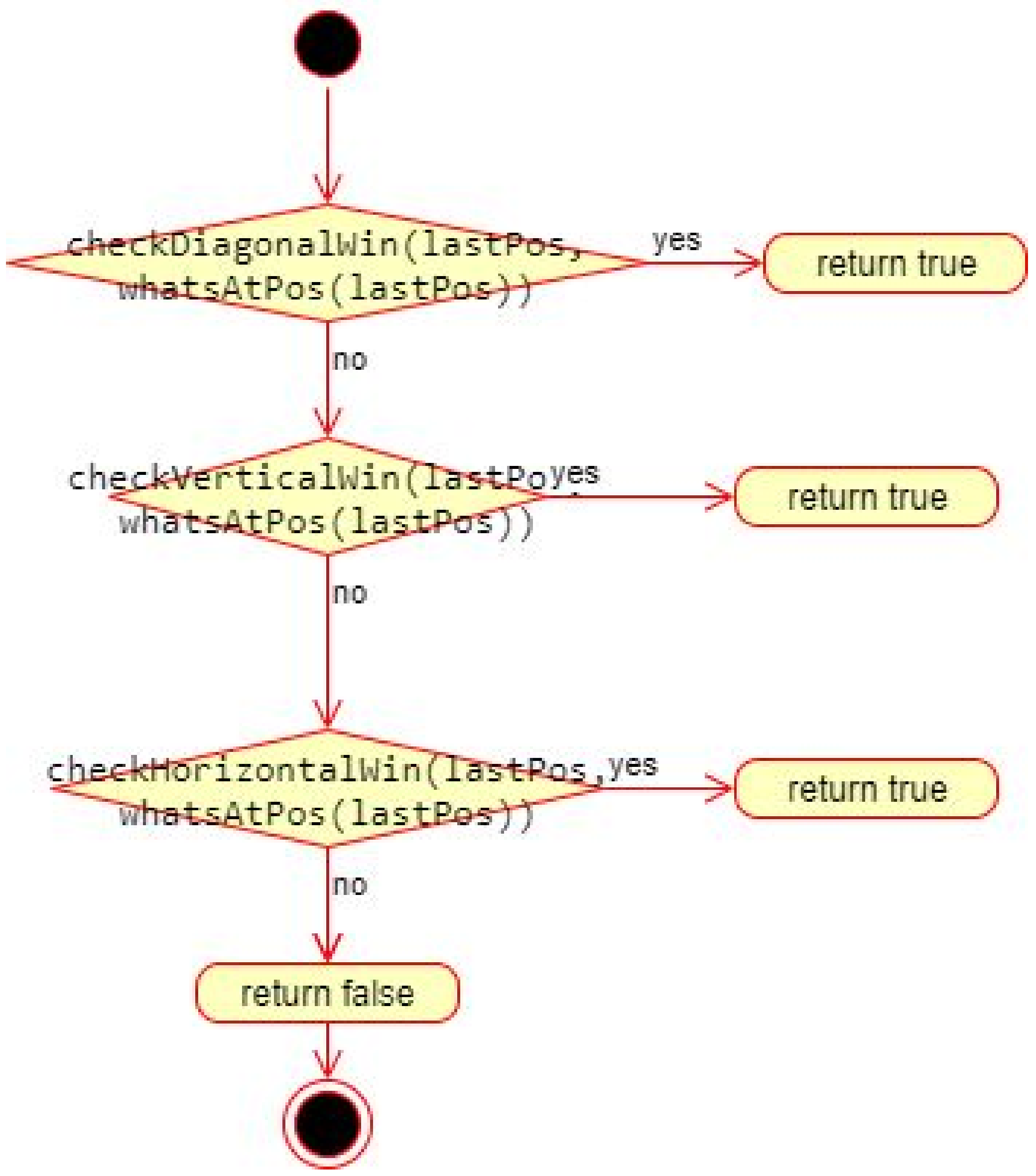
public boolean checkDiagonalWin(BoardPosition lastPos, char player)


countDownLeft = 0; countUpRight = 0; countDownRight = 0; countUpLeft = 0;
newPos = new BoardPosition(lastPos.getRow() - countDownLeft,lastPos.getColumn() - countDownLeft);
isPlayerAtPos(newPos, player)
no
yes
countDownLeft++
newPos = new BoardPosition(lastPos.getRow() + countUpRight,lastPos.getColumn() + countUpRight);
isPlayerAtPos(newPos, player)
no
yes
countUpRight++
countDownLeft + countUpRight - 1 >= getNumToWin()
no
yes
return true
newPos = new BoardPosition(lastPos.getRow() - countDownRight,lastPos.getColumn() + countDownRight);
isPlayerAtPos(newPos, player)
yes
countDownRight++
newPos = new BoardPosition(lastPos.getRow() + countUpLeft,lastPos.getColumn() - countUpLeft);
isPlayerAtPos(newPos, player)
yes
no
countUpLeft++
countDownRight + countUpLeft - 1 >= getNumToWin()
yes
no
return true
return false

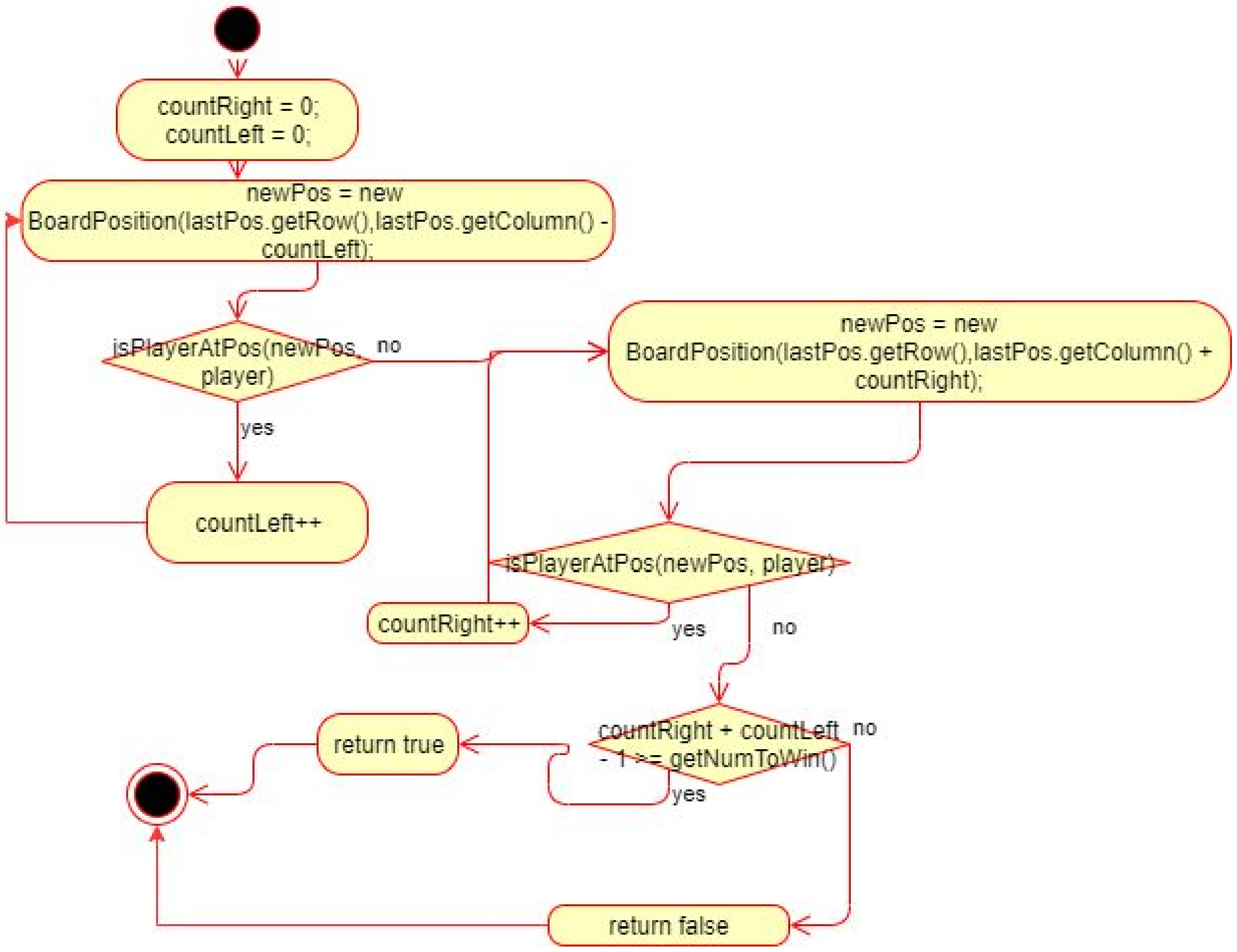
public boolean checkForDraw()



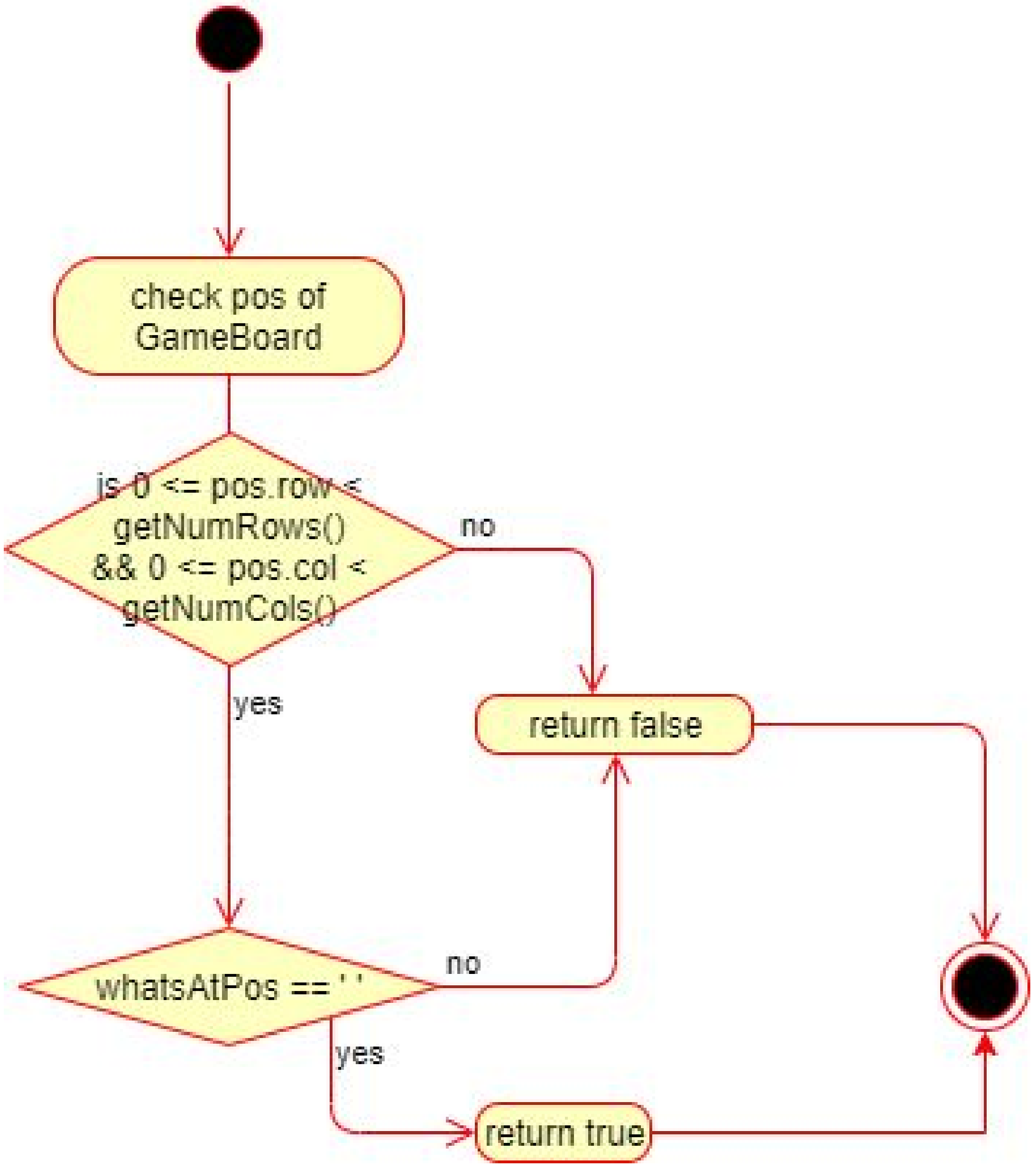
public boolean checkForWinner(BoardPosition lastPos)


checkDiagonalWin(lastPos, whatsAtPos(lastPos))
yes
return true
no
checkVerticalWin(lastPos, whatsAtPos(lastPos))
yes
return true
no
checkHorizontalWin(lastPos, whatsAtPos(lastPos))
yes
return true
no
return false

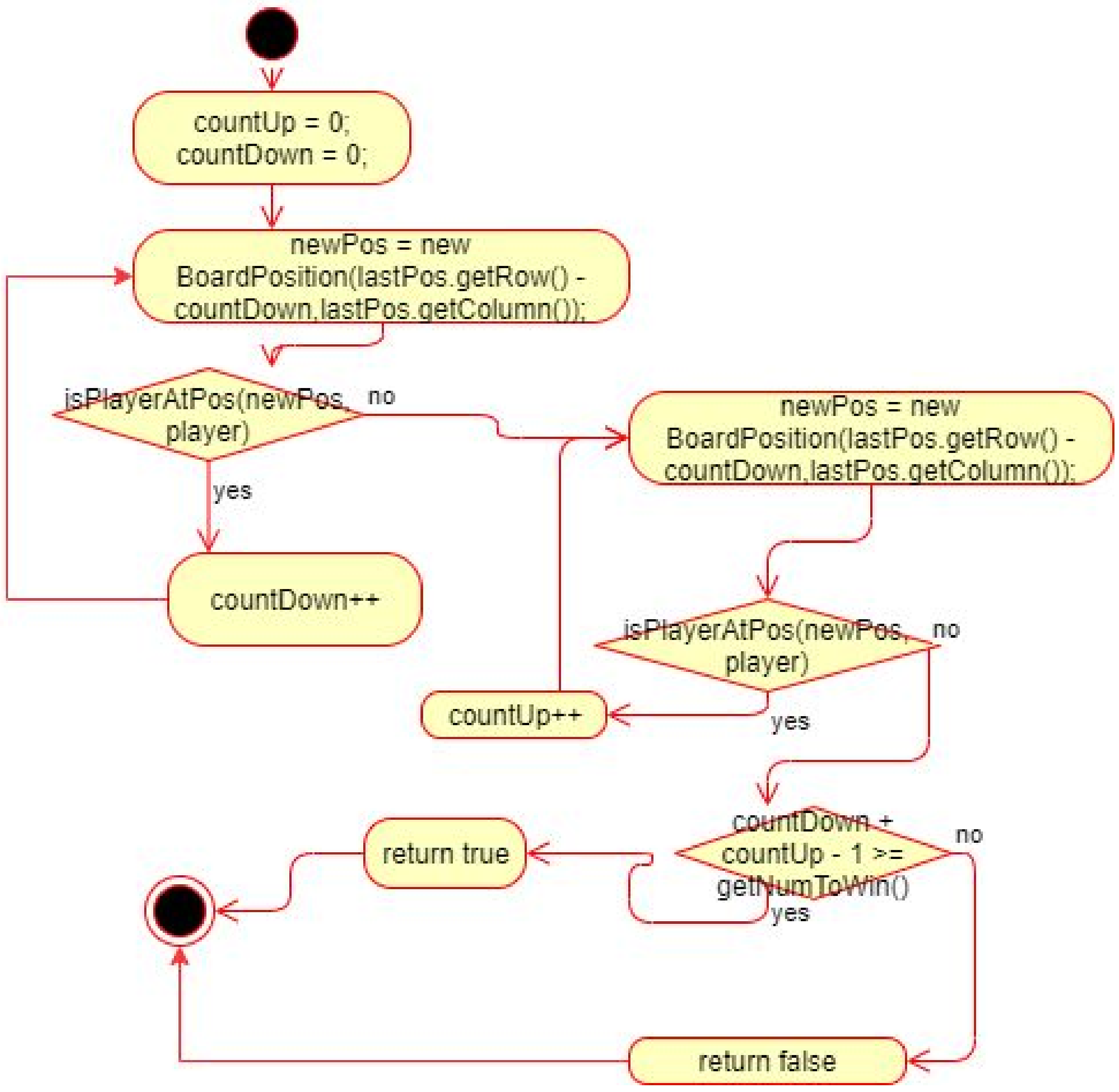
public boolean checkHorizontalWin(BoardPosition lastPos, char player)


countRight = 0;
countLeft = 0;
newPos = new BoardPosition(lastPos.getRow(),lastPos.getColumn() - countLeft);
isPlayerAtPos(newPos, player)
no
yes
countLeft++
newPos = new BoardPosition(lastPos.getRow(),lastPos.getColumn() + countRight);
isPlayerAtPos(newPos, player)
countRight++
yes
no
countRight + countLeft - 1 >= getNumToWin()
no
yes
return true
return false

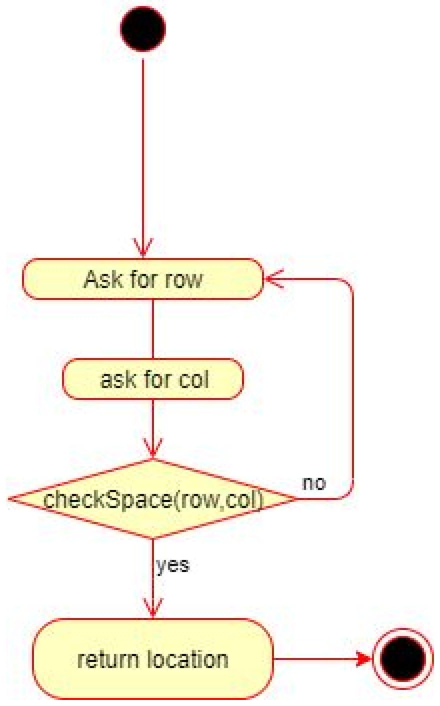
public boolean checkSpace(BoardPosition pos)



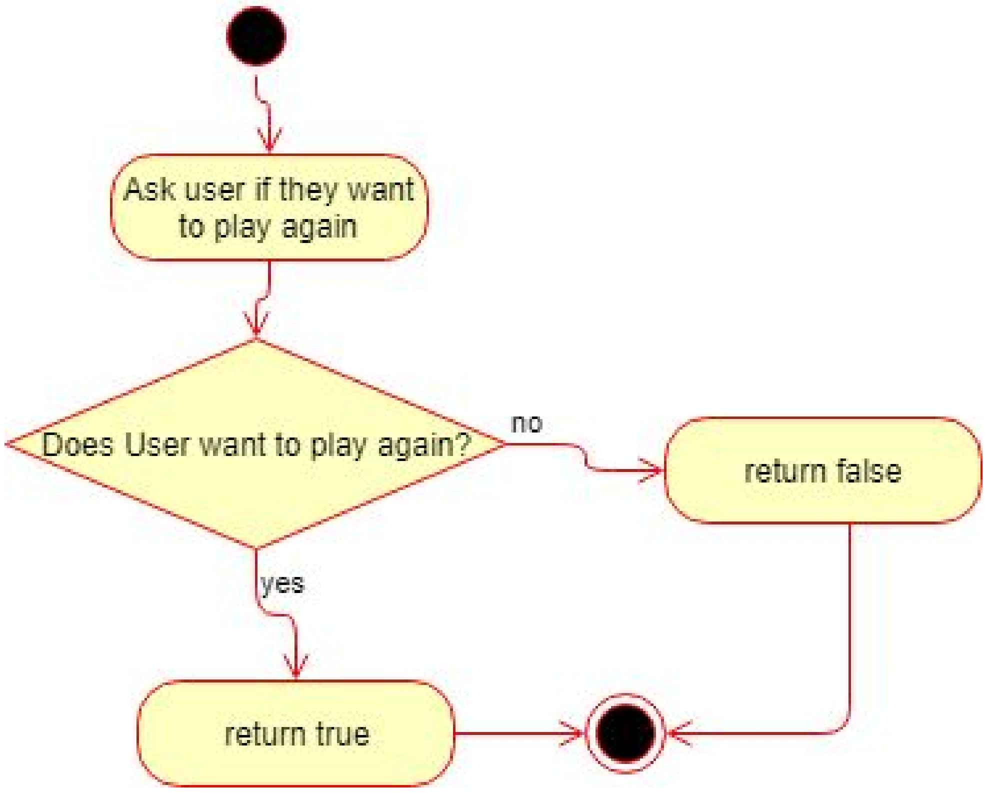
public boolean checkVerticalWin(BoardPosition lastPos, char player)


countUp = 0;
countDown = 0;
newPos = new BoardPosition(lastPos.getRow() - countDown,lastPos.getColumn());
isPlayerAtPos(newPos, player)
no
yes
countDown++
newPos = new BoardPosition(lastPos.getRow() - countDown,lastPos.getColumn());
isPlayerAtPos(newPos, player)
no
yes
countUp++
countDown + countUp - 1 >= getNumToWin()
no
yes
return true
return false

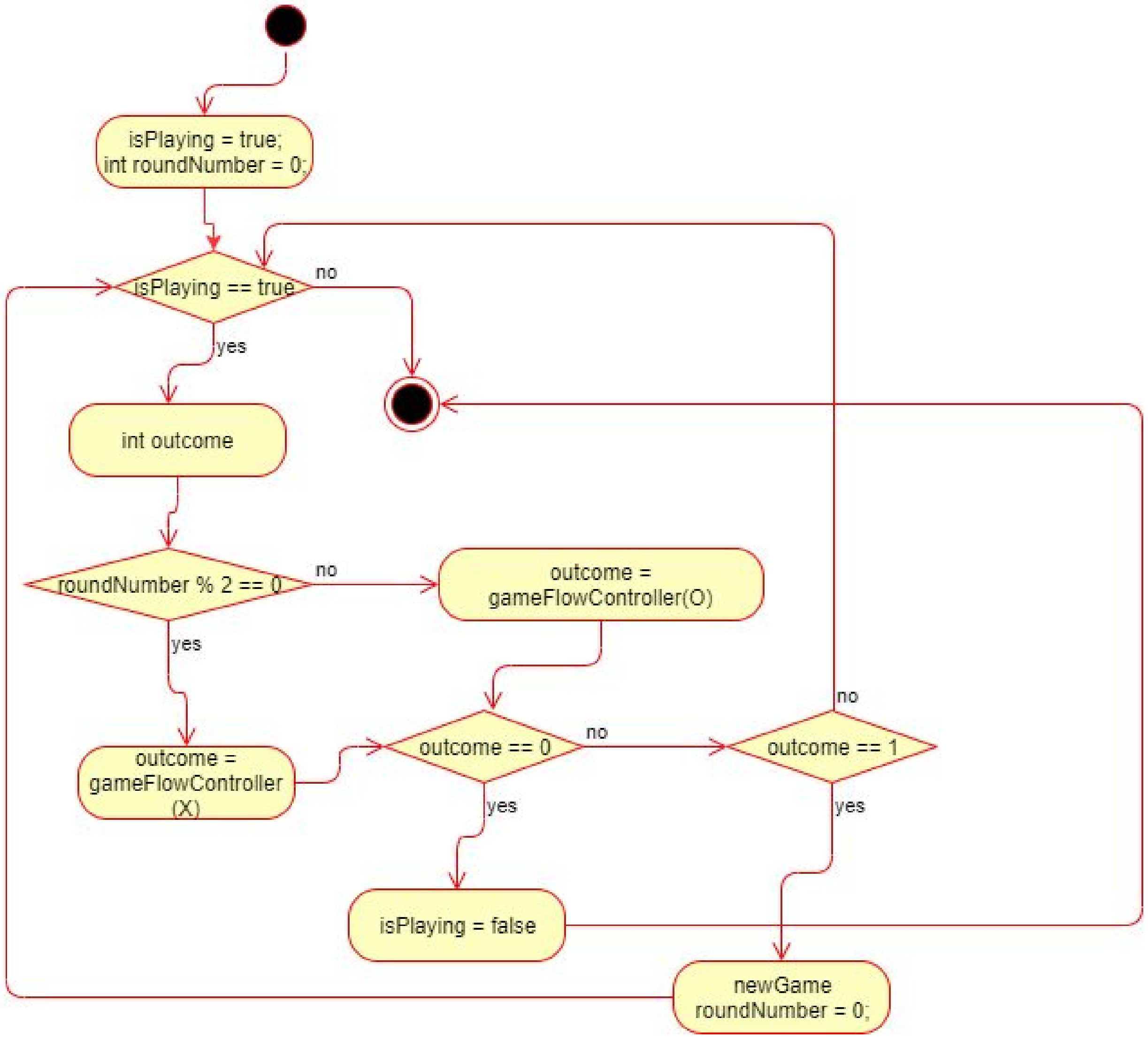
private static BoardPosition getLocationInput(IGameBoard Board, char player)



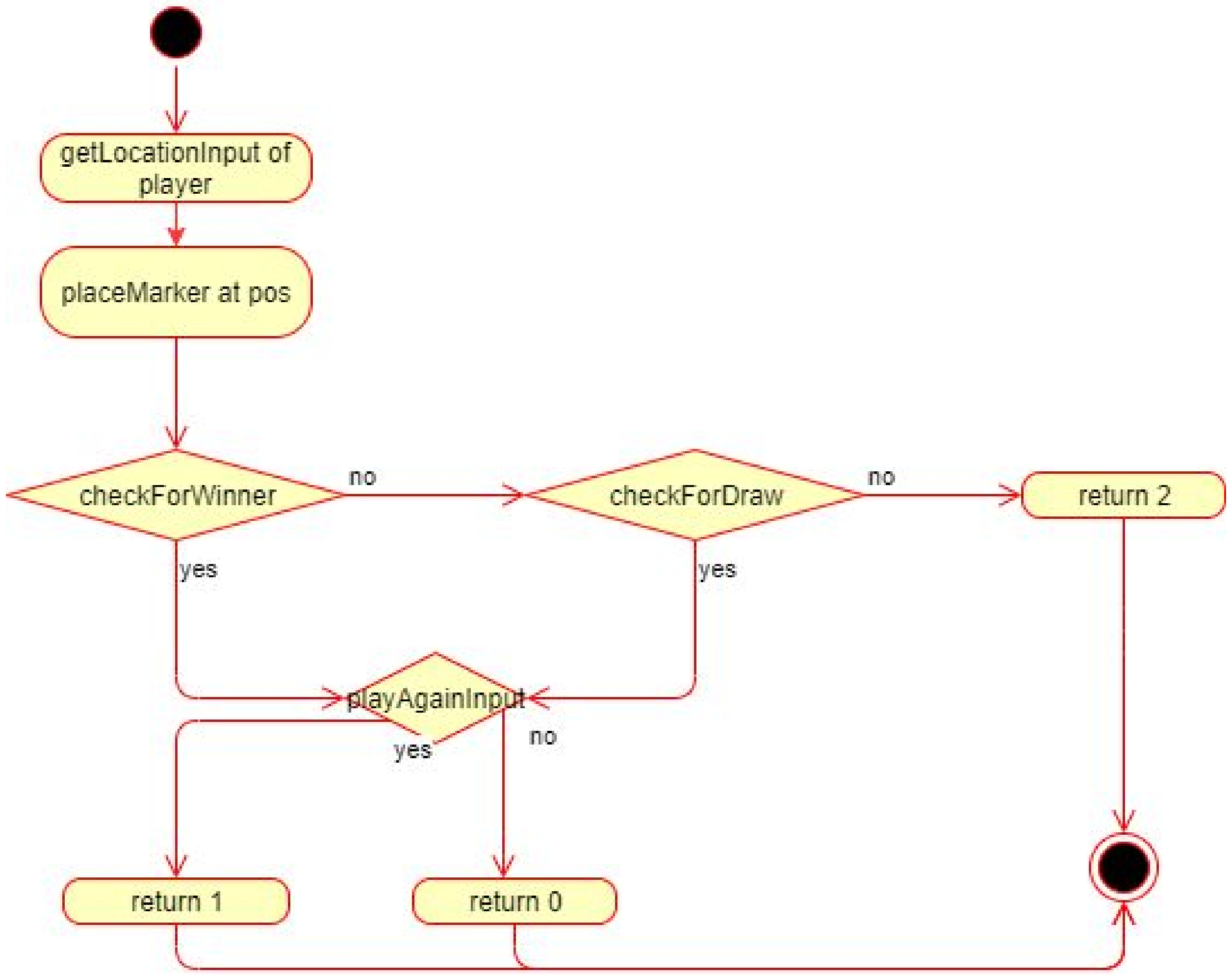
private static boolean getPlayAgainInput(IGameBoard Board, char player)


Ask user if they want to play again
Does User want to play again?
no
return false
yes
return true

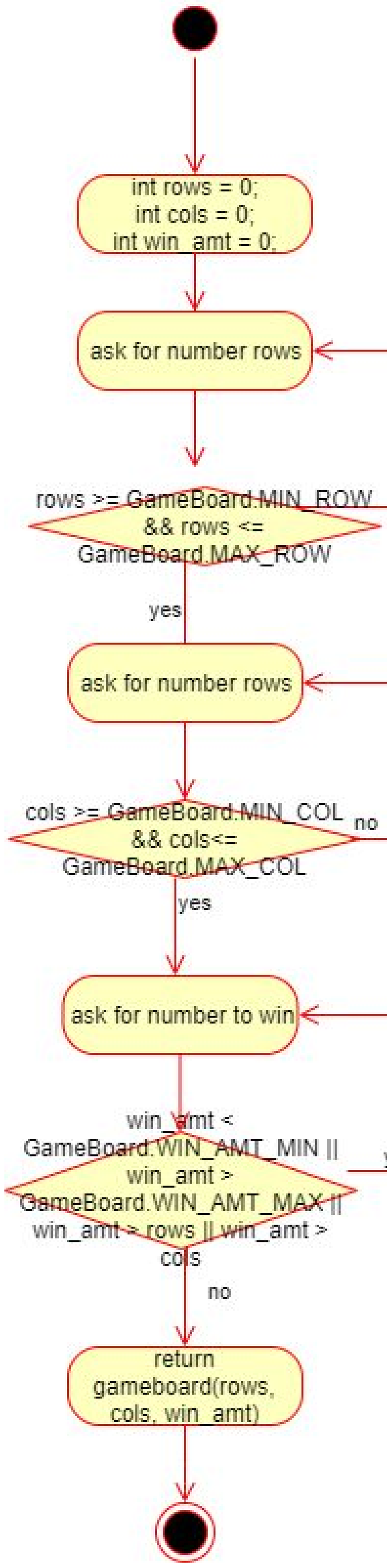
public static void main(String args[])



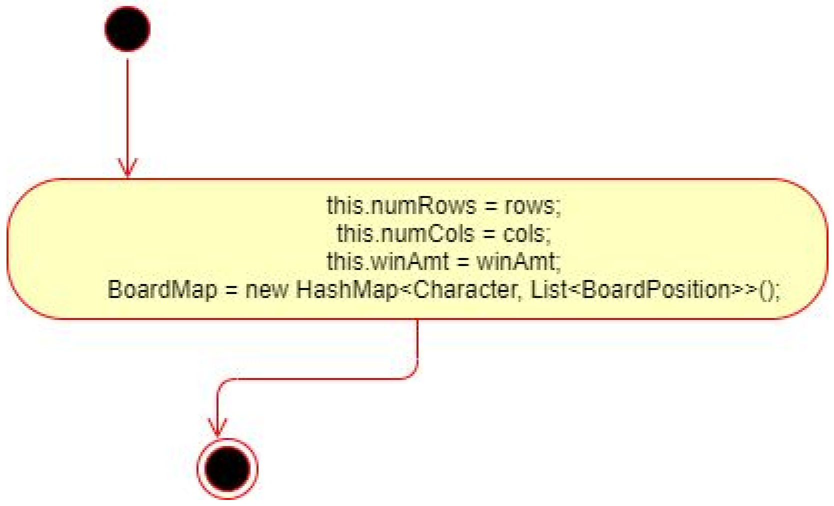
private static int gameFlowController(IGameBoard newGame, char player)



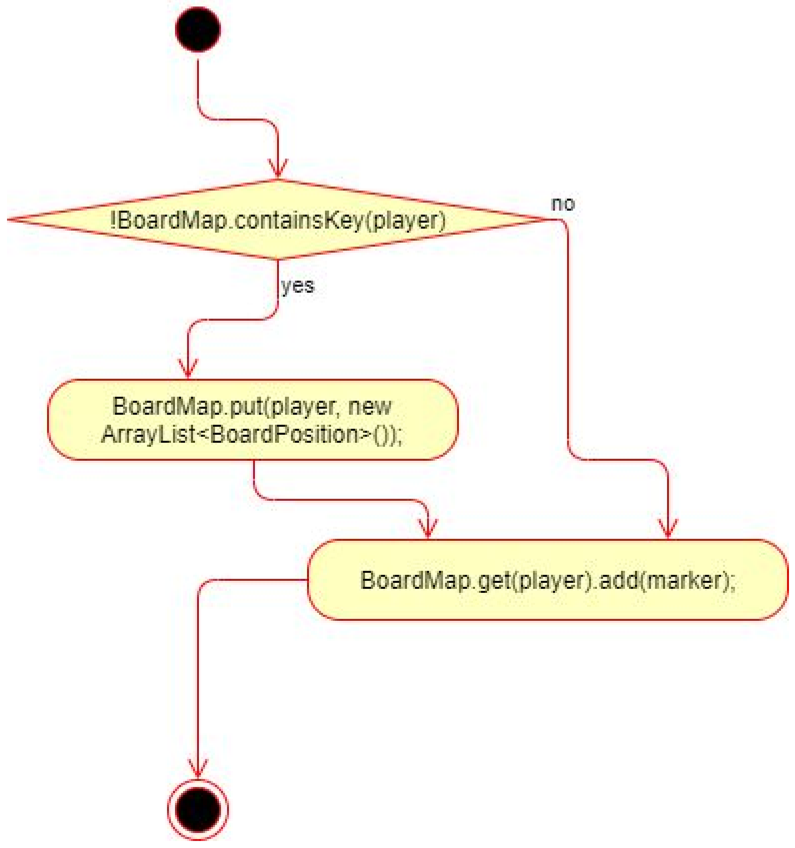
private static IGameBoard makeNewGameBoard()


int rows = 0;
int cols = 0;
int win_amt = 0;
ask for number rows
no
rows >= GameBoard.MIN_ROW && rows <= GameBoard.MAX_ROW
yes
ask for number rows
cols >= GameBoard.MIN_COL && cols<= GameBoard.MAX_COL
no
yes
ask for number to win
win_amt < GameBoard.WIN_AMT_MIN || win_amt > GameBoard.WIN_AMT_MAX || win_amt > rows || win_amt > cols
yes
no
return gameboard(rows, cols, win_amt)

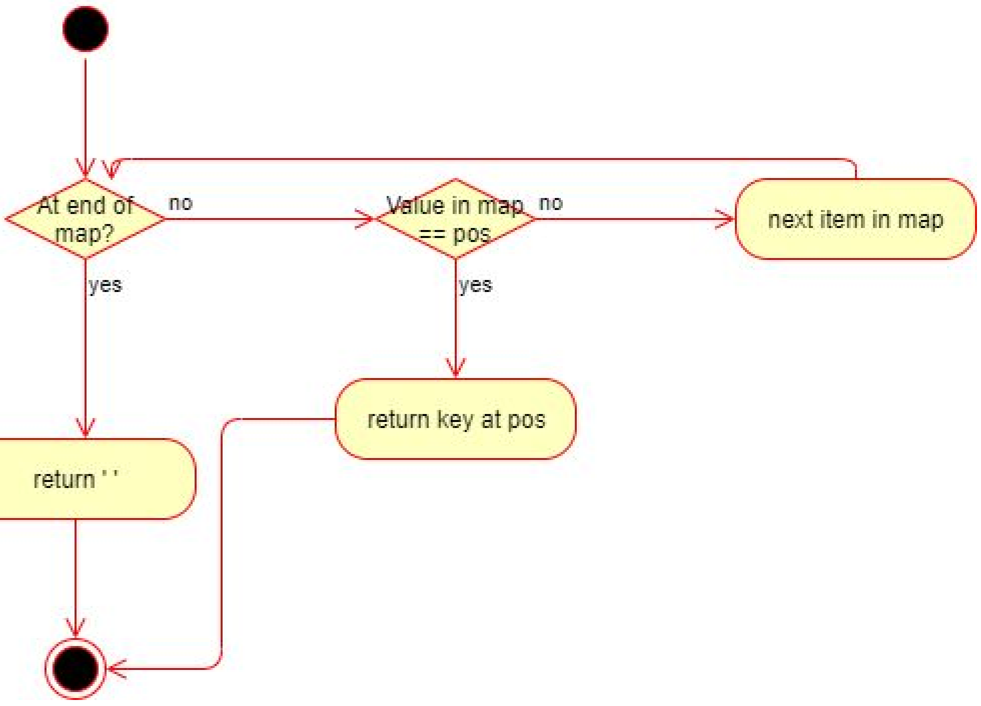
public GameBoardMem(int rows, int cols, int winAmt, List<Character> charList)



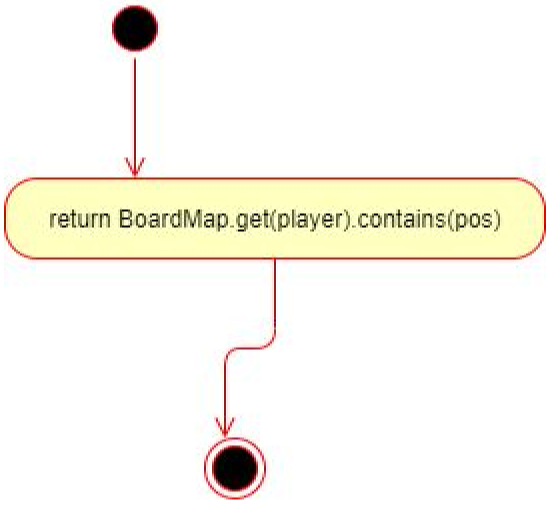
(GamBoardMem) public void placeMarker(BoardPosition marker, char player)


!BoardMap.containsKey(player)
no
yes
BoardMap.put(player, new ArrayList<BoardPosition>());
BoardMap.get(player).add(marker);

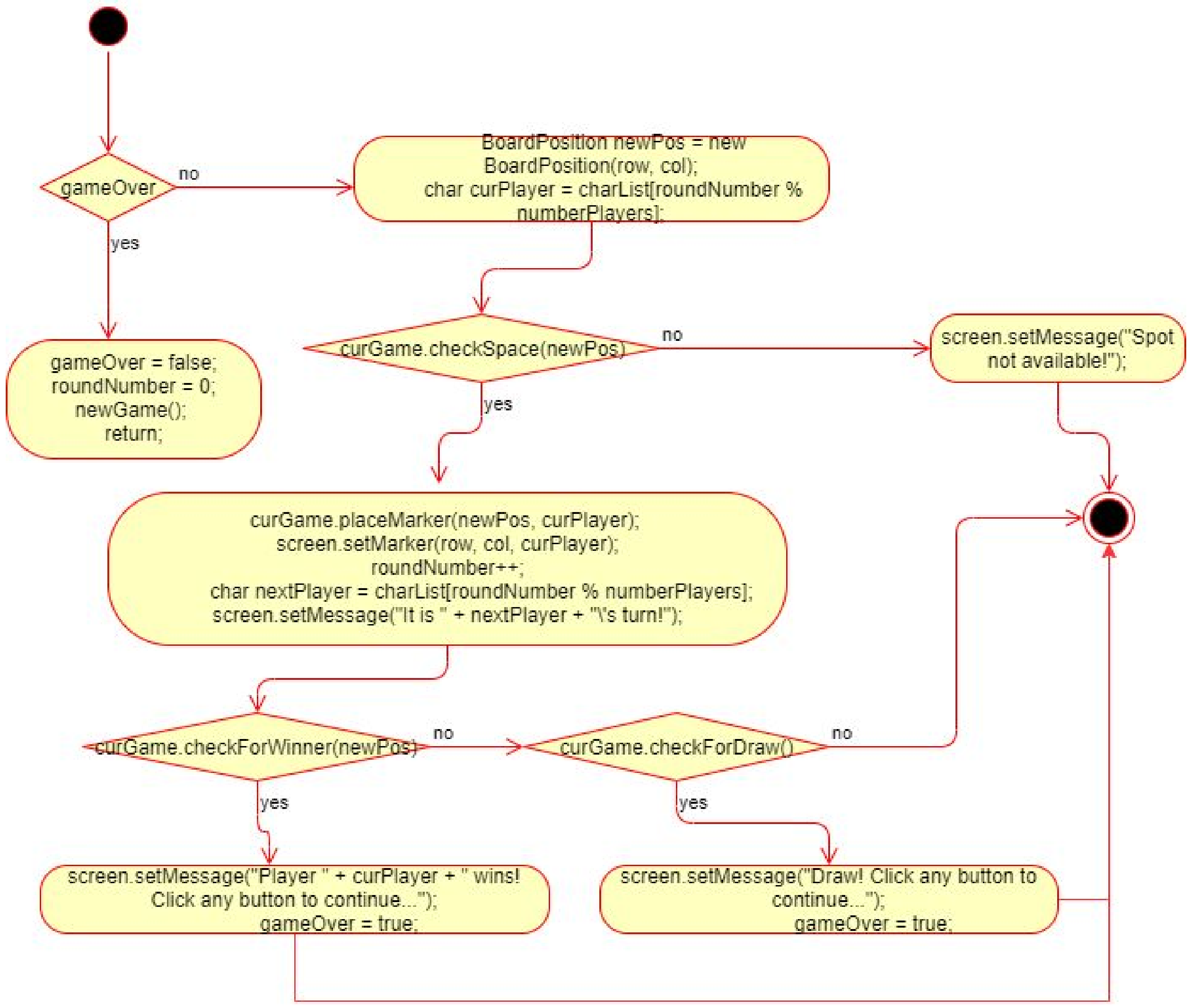
(GamBoardMem) public char whatsAtPos(BoardPosition pos)



(GamBoardMem) public boolean isPlayerAtPos(BoardPosition pos, char player)



public void processButtonClick(int row, int col)



gameOver
no
yes
BoardPosition newPos = new BoardPosition(row, col);
char curPlayer = charList[roundNumber % numberPlayers];
gameOver = false;
roundNumber = 0;
newGame();
return;
curGame.checkSpace(newPos)
no
yes
screen.setMessage("Spot not available!");
curGame.placeMarker(newPos, curPlayer);
screen.setMarker(row, col, curPlayer);
roundNumber++;
char nextPlayer = charList[roundNumber % numberPlayers];
screen.setMessage("It is " + nextPlayer + "\'s turn!");
curGame.checkForWinner(newPos)
no
yes
curGame.checkForDraw()
no
yes
screen.setMessage("Player " + curPlayer + " wins! Click any button to continue...");
gameOver = true;
screen.setMessage("Draw! Click any button to continue...");
gameOver = true;