CPSC 2150 Project Report

Carson Tollison

**Requirements Analysis**

**Functional Requirements:**

1. As a player, I need be able to see the board so that I know which positions are available on the game board.
2. As a player, I need to know which marker I use so that I know who’s turn it is.
3. As a player, I need to be able to input my column number so that I can put my marker into that position.
4. As a player, I need to know how the columns are labeled so that I place my marker into the correct position.
5. As a player, I need to be able to choose if I want to play again or not so that I can quit or continue playing.
6. As a player, I can input my response again to play again if my input was invalid.
7. As a player, I can input my column again if I have invalid input so that I get my marker on the board.
8. As a player, I can input the number of columns I want to use on this game board.
9. As a player, I can input my number of columns again so that it is within the correct range.
10. As a player, I can input the number of rows I want to use on this game board.
11. As a player, I can input my number of rows again so that it is within the correct range.
12. As a player, I can input the number of tokens in a row it takes to win the game.
13. As a player, I can input my number of tokens in a row again so that it is within the correct range.
14. As a player, I need to know the game status so that I will know if it was a win, tie, or loss.
15. As a player, if I get the number of tokens in a row horizontally that I specified, I will win the game so that I can win the game
16. As a player, if I get the number of tokens in a row vertically that I specified, I will win the game so that I can win the game
17. As a player, if I get the number of tokens in a row diagonally that I specified, I will win the game so that I can win the game
18. As a player, I can pick again if I pick a column that does not exist, so I don't lose my turn
19. As a player, I can make a move after my opponent does (assuming they don't win), so I can always have my turn
20. As a player, I can end the game in a tie by taking the last space on the board without getting the number of tokens in a row that I specified, so the game can end
21. As a player, I can input the number of players that will play the game.
22. As a player, I can input the number of players again if the input was invalid.
23. As a player, I can input the token that each player will use.
24. As a player, I can input the token that a player would use again if I inputted a token that is taken.
25. As a player, I can choose if I want to play a fast game or a memory efficient game.
26. As a player, I can input my choice for a game type again if my input was invalid.

**Non-Functional Requirements**

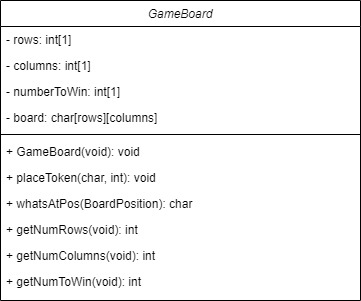
1. The system must be programmed with java.
2. The system must run on Unix.
3. The system should be reliable.
4. The system should be fast if that game mode is chosen.
5. The system should be memory efficient if that game mode is chosen.
6. The board can be any size that the user specifies if it falls within range.

**Deployment Instructions**default: compiles code. Runs with the *make* command.   
run: runs code. Runs with the *make run* command.   
clean: removes compiled (.class) files. Runs with the *make clean* command.

**System Design**

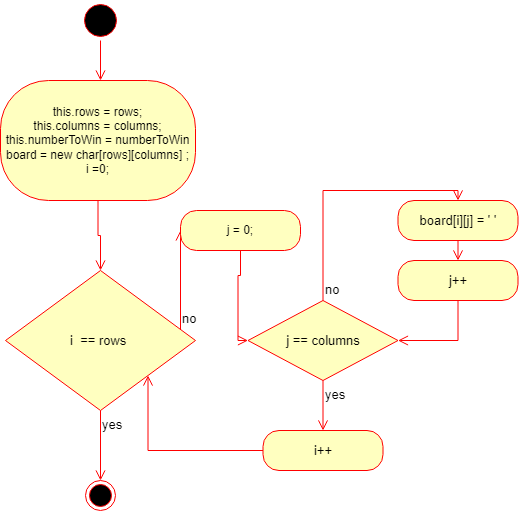
**Class 1:** GameBoard.java

**Class diagram**

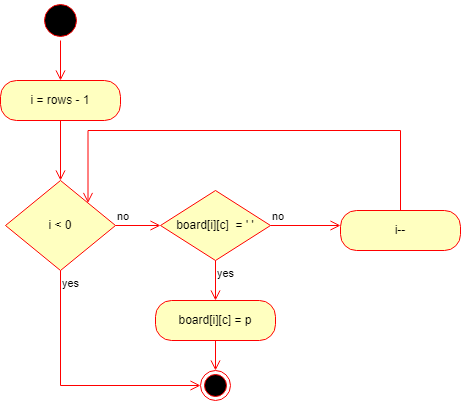
****

**Activity diagrams**

**GameBoard**

****

**placeToken**

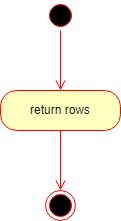
****

**whatsAtPos**

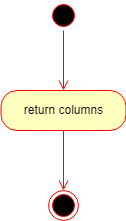
**Diagram

Description automatically generated**

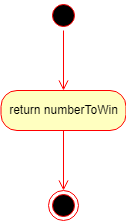
**getNumRows**

****

**getNumColumns**

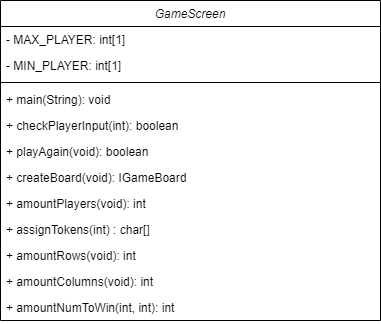
****

**getNumToWin**

****

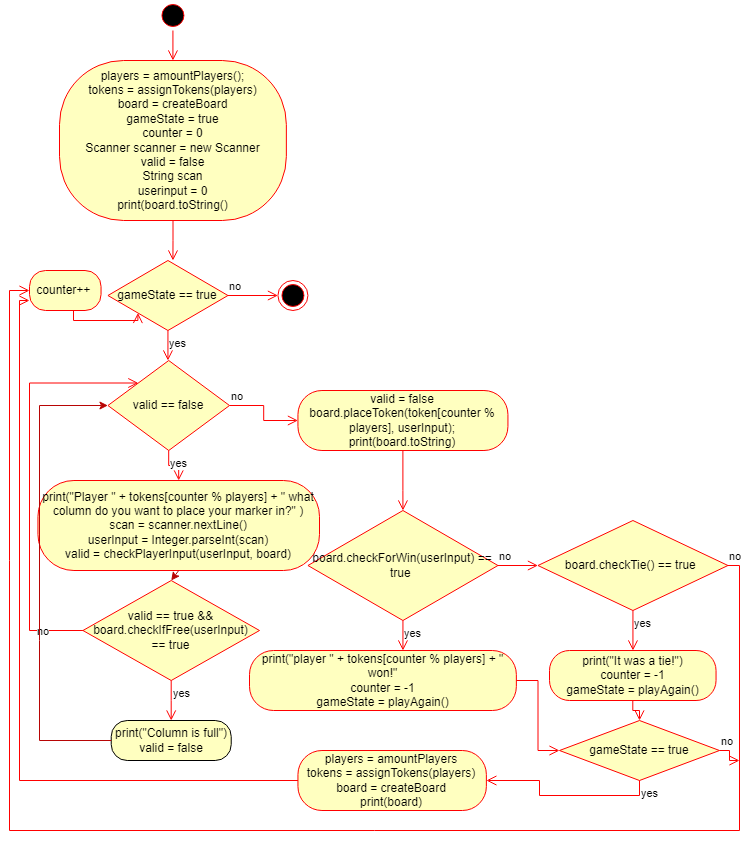
**Class 2:** GameScreen.java

**Class diagram**

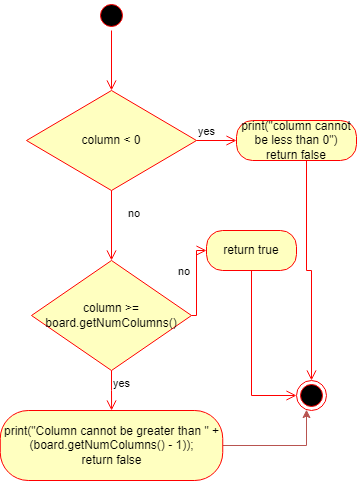
****

**Activity diagrams**

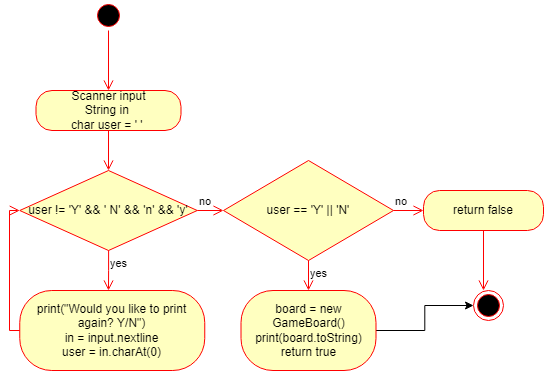
**main**

****

**checkPlayerInput**

****

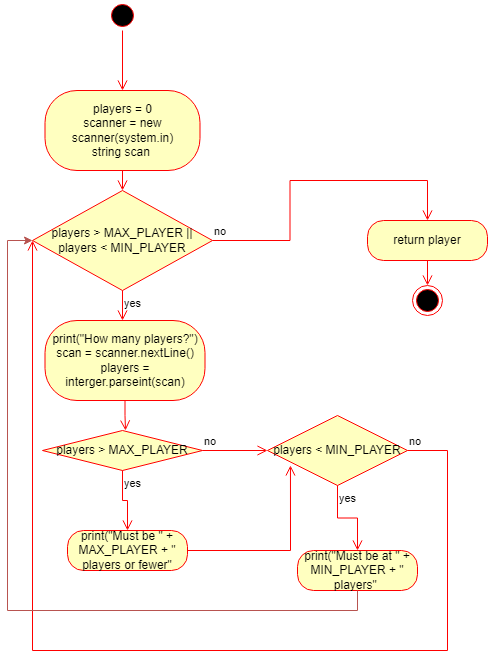
**playAgain**

****

**createBoardDiagram

Description automatically generated**

**amountPlayers**

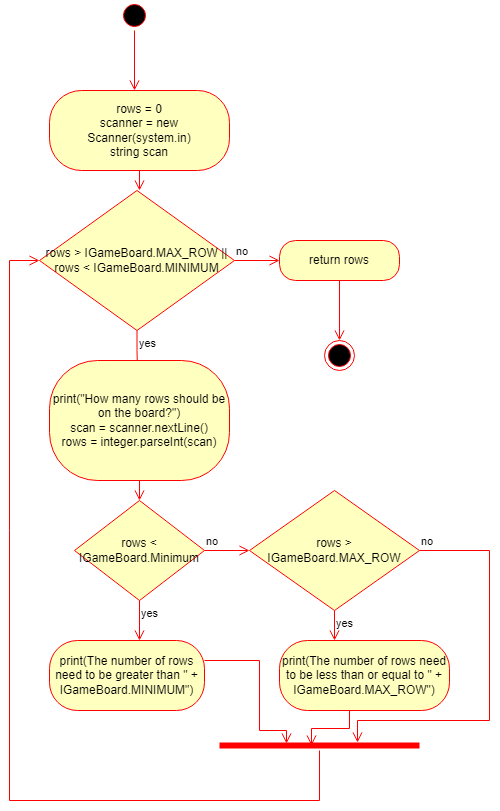
****

**assignTokens**

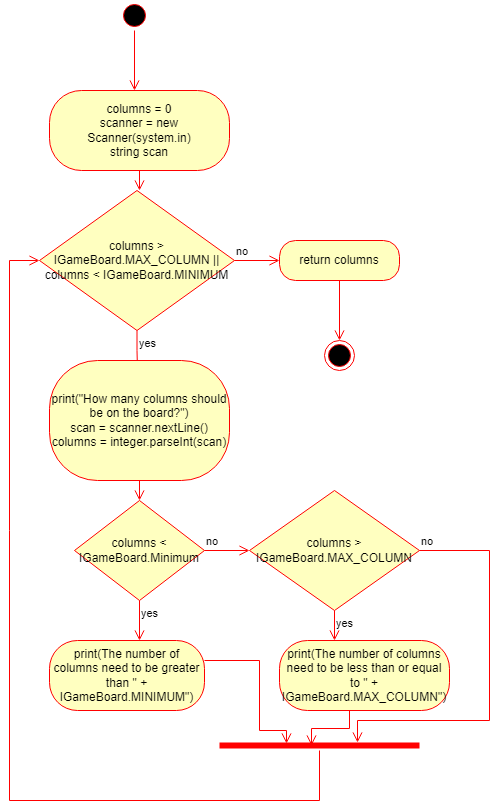
**Diagram

Description automatically generated**

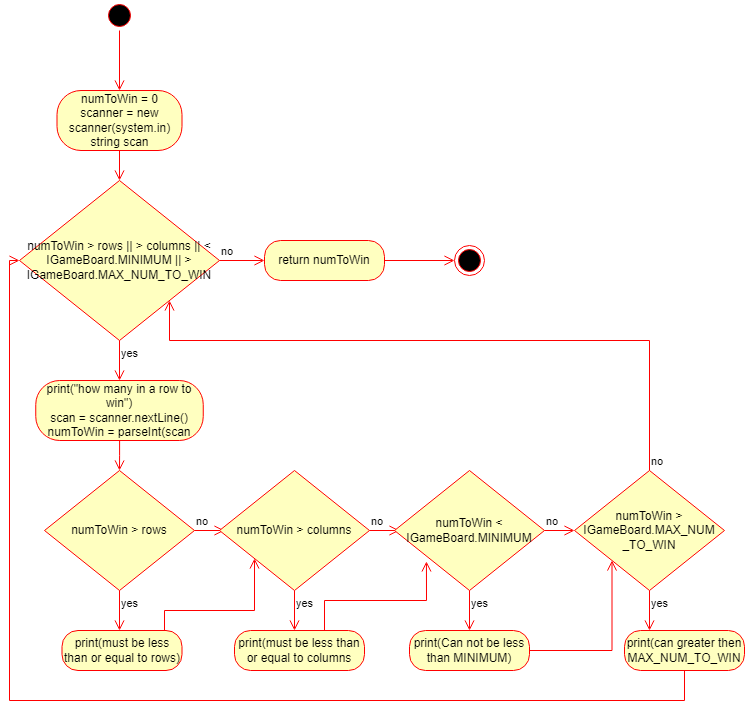
**amountRows**

****

**amountColumns**

****

**amountNumToWin**

****

**Class 3:** BoardPosition.java

**Class diagram**

**Table

Description automatically generated**

**Activity diagrams**

**BoardPosition**

**Diagram

Description automatically generated**

**getRow**

**Diagram

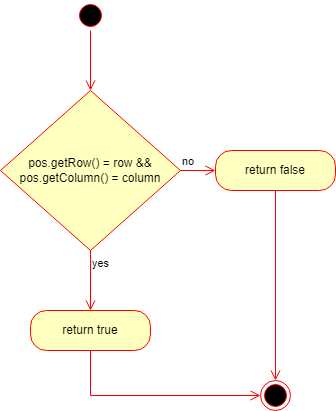
Description automatically generated**

**getColumn**

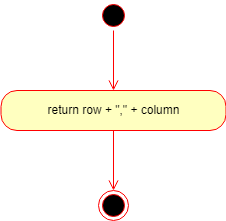
**Diagram

Description automatically generated**

**equals**

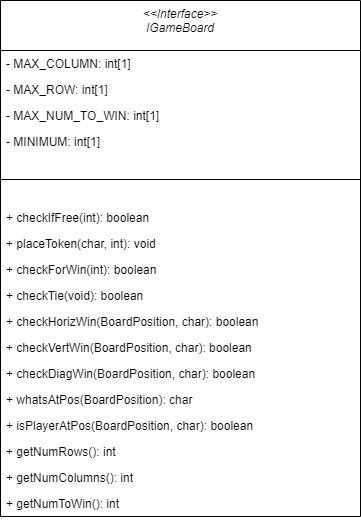
****

**toString**

****

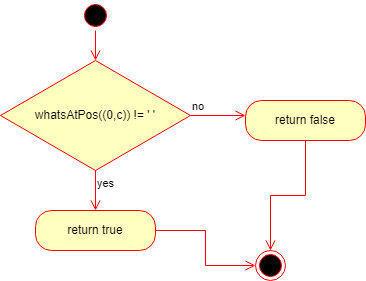
**Class 4:** IGameBoard.java

**Class diagram**

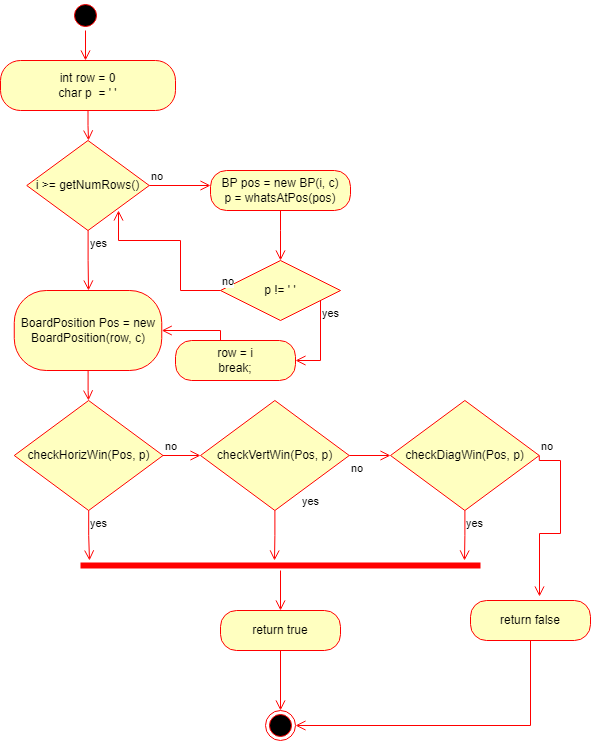
****

**Activity diagrams**

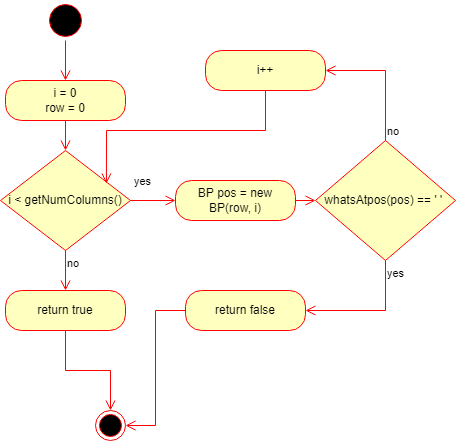
**checkIfFree**

****

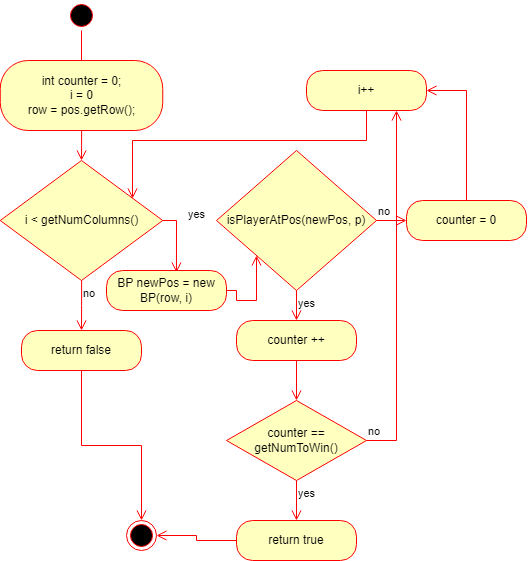
**checkForWin**

****

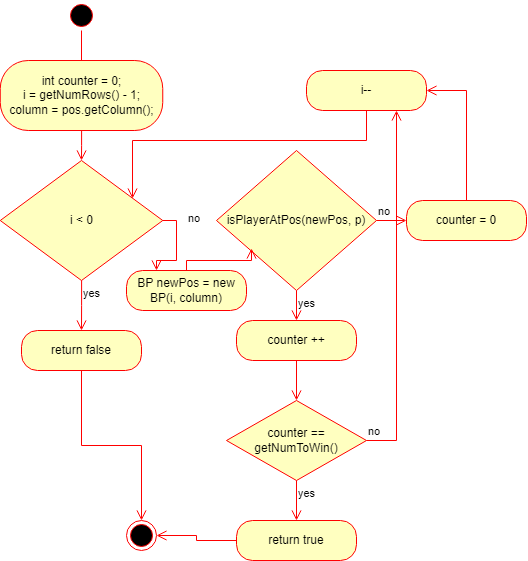
**checkTie**

****

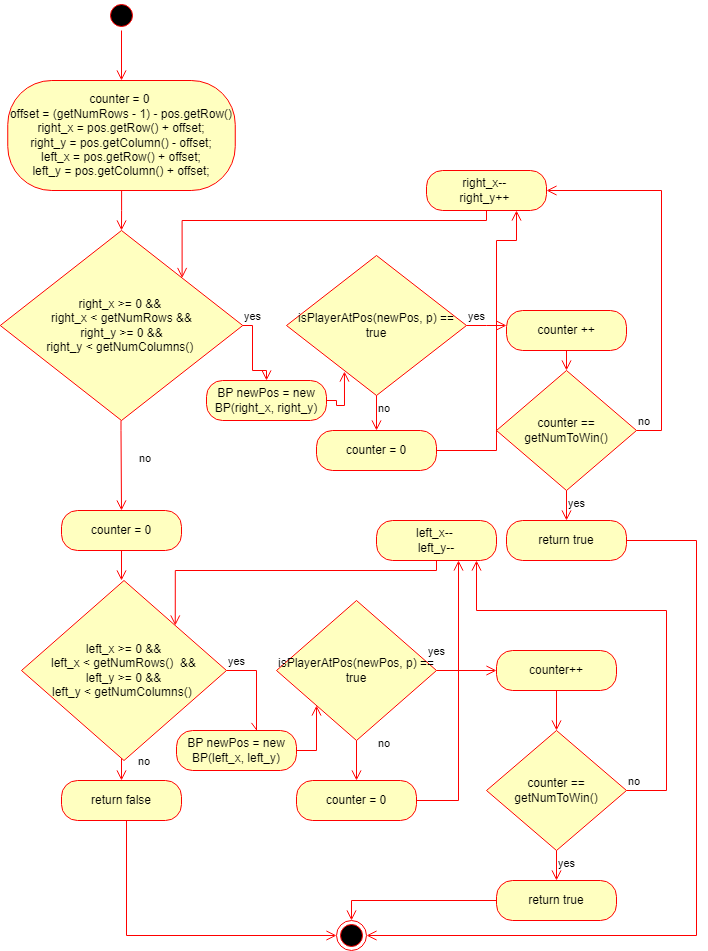
**checkHorizWin**

****

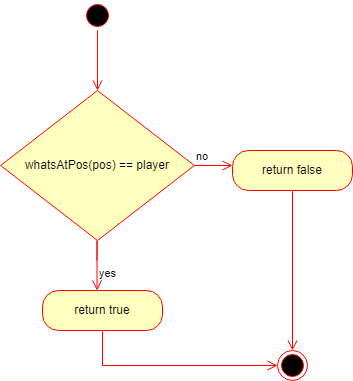
**checkVertWin**

****

**checkDiagWin**

****

**isPlayerAtPos**

****

**Class 5:** AbsGameBoard.java

**Class diagram**

**Table

Description automatically generated**

**Activity diagrams**

**toString**

**Diagram

Description automatically generated**

**Class 6:** GameBoardMem.java

**Class diagram**

**Table

Description automatically generated with medium confidence**

**Activity diagrams**

**GameBoardMem**

**Diagram

Description automatically generated**

**placeToken**

**Diagram

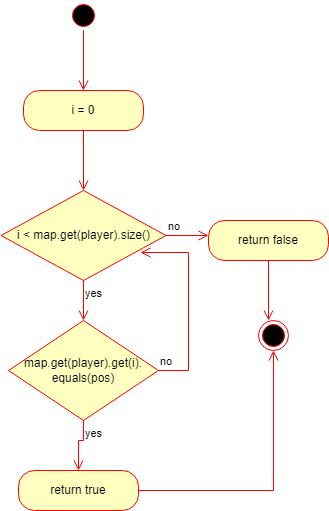
Description automatically generated**

**whatsAtPos**

**Diagram

Description automatically generated**

**isPlayerAtPos**

****

**getNumRows**

**Diagram

Description automatically generated**

**getNumColumns**

**Diagram

Description automatically generated**

**getNumToWin**

**Diagram

Description automatically generated**

**Test Cases**

Details in Project 4.