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 column again if I have invalid input so that I get my marker on the board.
7. As a player I need to know the game status so that I will know if it was a win, tie, or loss.
8. As a player, if I get 5 in a row horizontally, I will win the game so that I can win the game
9. As a player, if I get 5 in a row vertically, I will win the game so that I can win the game
10. As a player, if I get 5 in a row diagonally, I will win the game so that I can win the game
11. As a player, I can pick again if I pick a column that does not exist, so I don't lose my turn
12. As a player, I can make a move after my opponent does (assuming they don't win), so I can always have my turn
13. As a player, I can end the game in a tie by taking the last space on the board without getting 5 in a row, so the game can end

**Non-Functional Requirements**

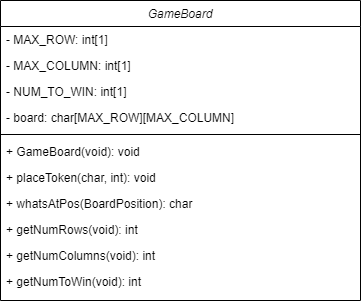
1. The system must be programmed with java.
2. The system must run on Unix.
3. The system should be reliable.
4. The system should be fast.
5. Board is of size 6 x 9

**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**

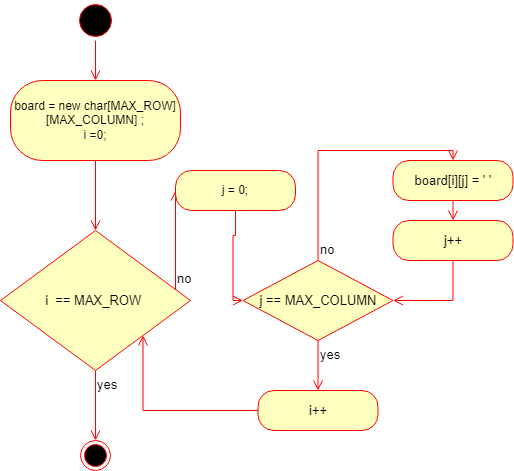
**Class 1:** GameBoard.java

**Class diagram**

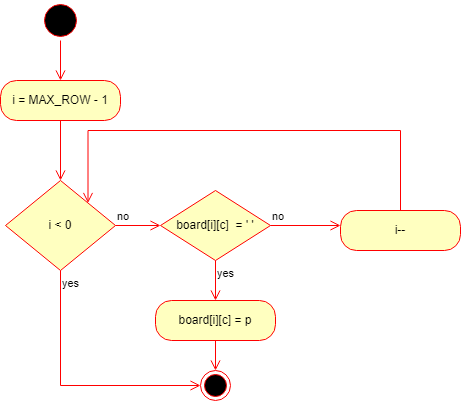
****

**Activity diagrams**

**GameBoard**

****

**placeToken**

****

**whatsAtPos**

**Diagram

Description automatically generated**

**getNumRows**

**Diagram

Description automatically generated**

**getNumColumns**

**Diagram

Description automatically generated**

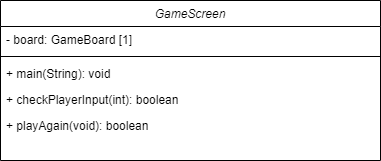
**getNumToWin**

**Diagram

Description automatically generated**

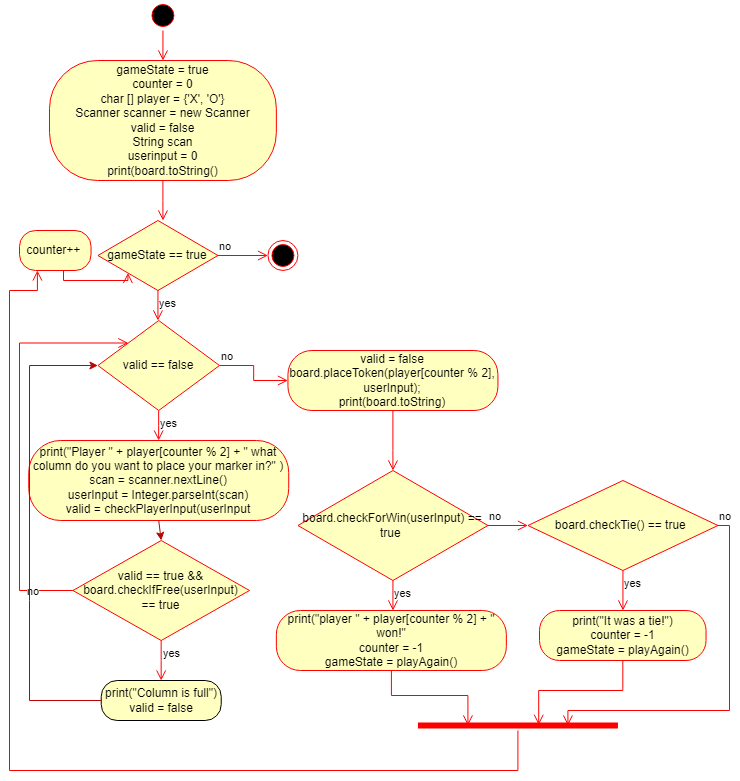
**Class 2:** GameScreen.java

**Class diagram**

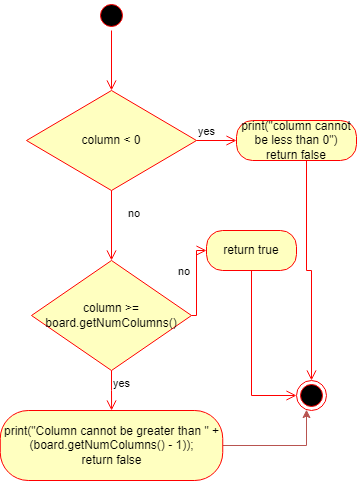
****

**Activity diagrams**

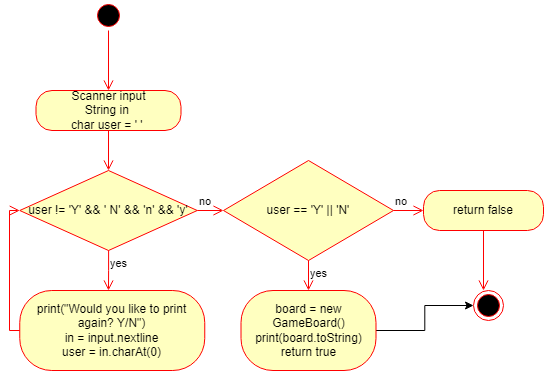
**main**

****

**checkPlayerInput**

****

**playAgain**

****

**Class 3:** BoardPosition.java

**Class diagram**

**Table

Description automatically generated**

**Activity diagrams**

**BoardPosition**

**Diagram

Description automatically generated**

**getRow**

**Diagram

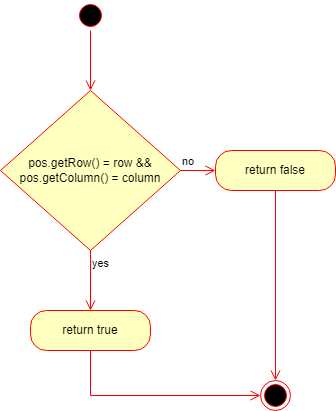
Description automatically generated**

**getColumn**

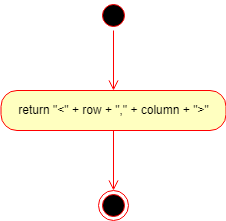
**Diagram

Description automatically generated**

**equals**

****

**toString**

****

**Class 4:** IGameBoard.java

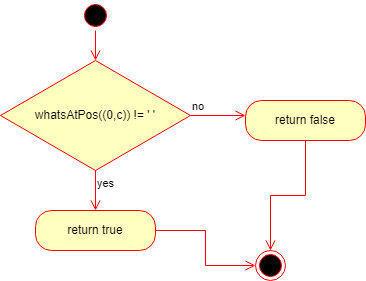
**Class diagram**

**Table

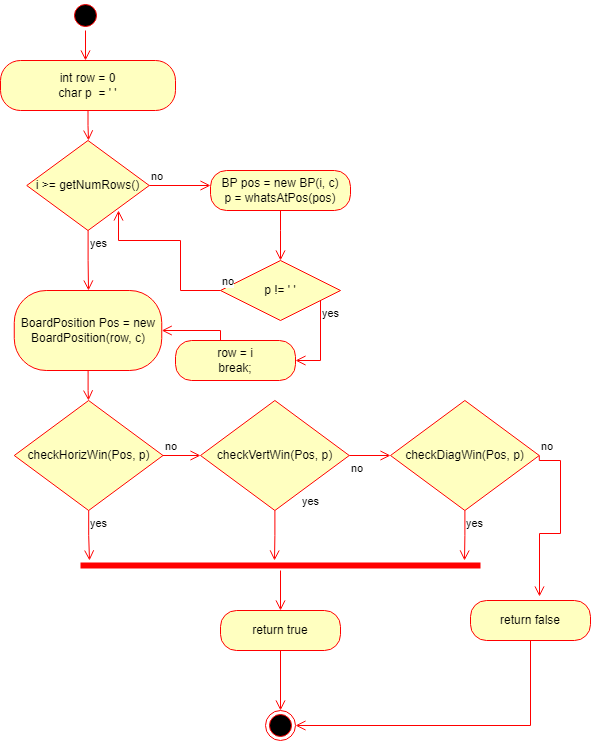
Description automatically generated**

**Activity diagrams**

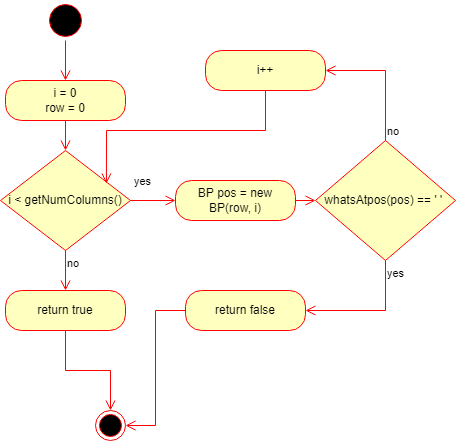
**checkIfFree**

****

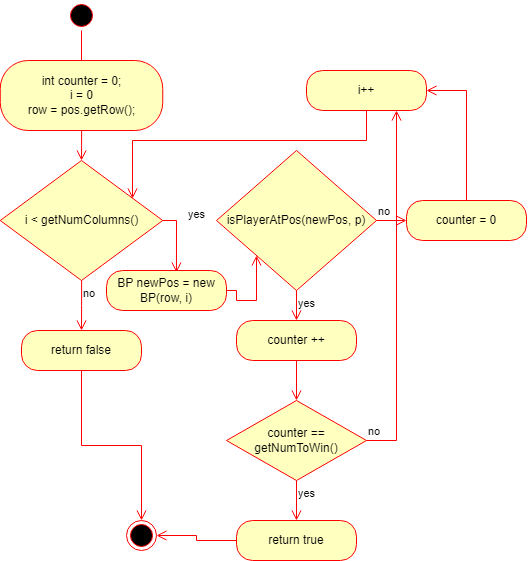
**checkForWin**

****

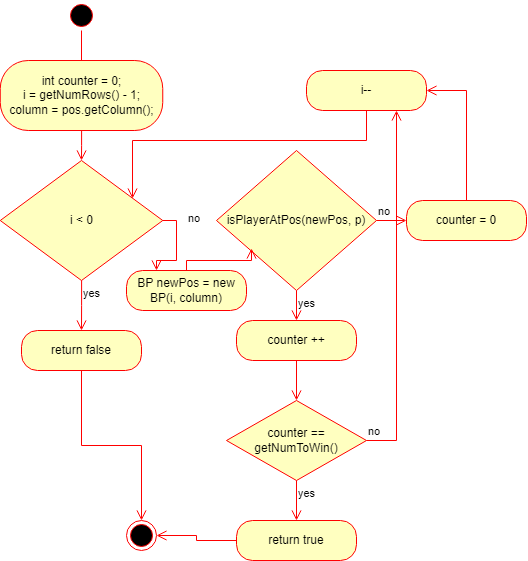
**checkTie**

****

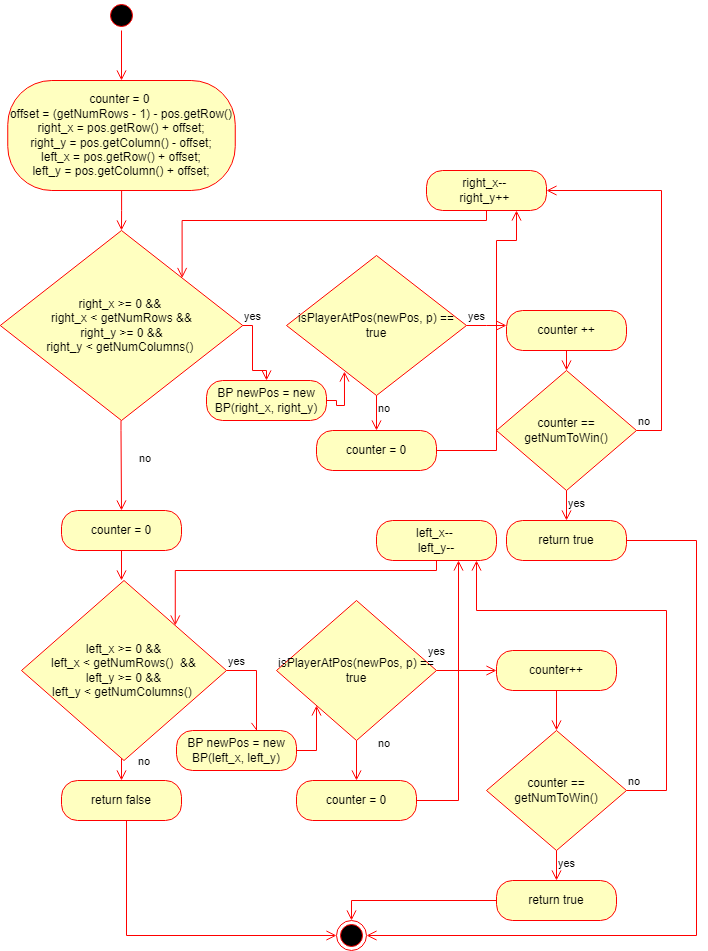
**checkHorizWin**

****

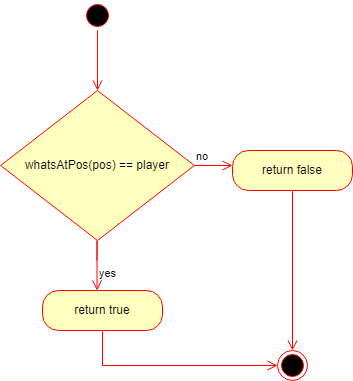
**checkVertWin**

****

**checkDiagWin**

****

**isPlayerAtPos**

****

**Class 5:** AbsGameBoard.java

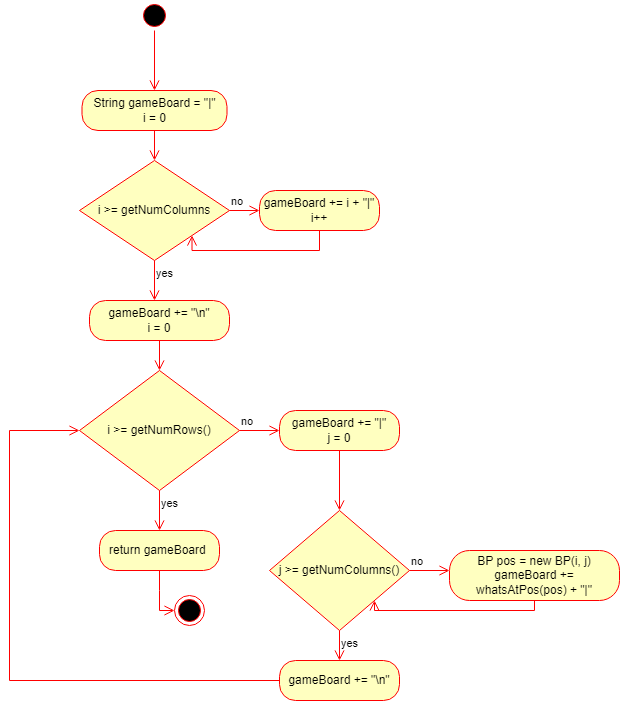
**Class diagram**

**Table

Description automatically generated**

**Activity diagrams**

**toString**

****

**Test Cases**

Details in Project 4.