Ashlyn Cooper

CPSC 2150

Project 3

Project Report

Requirements Analysis

Functional Requirements

1. As a player, I can input a position so that I can play my token.
2. As a player, I can choose to play again so that I can replay tic tac toe.
3. As a player, I can have my input validated by the program so that I can place my token in a valid position.
4. As a player, I can view output prompting for an input so that I can see when I should make an input.
5. As a player, I can view an outputted message at the end of the game, so that I know how the game ended.
6. As a player, I can win horizontally, as that I can win the game.
7. As a player, I can win vertically, so that I can win the game.
8. As a player, I can win diagonally so that I can win the game.
9. As a player, I can make a move after my opponent (if they have not one), so that I can take my turn.
10. The program can end the game without either player winning, so that the game can end in a tie.
11. As a user, I can input the number of columns and number of rows, so that I can set the size of the gameboard.
12. As a user, I can input the number in a row needed to win, so that the game is played with that rule.
13. As a user, I can input the number of players, so that the game can be played with that number of players.
14. As a player, I can input a unique character, so that I can have a unique token as my marker.
15. As a user, I can choose between a memory efficient or fast implementation of the game, so that the program runs with the desired implementation.

Non-functional Requirements

1. The system must be written in Java.
2. The system must run on a Unix machine.
3. The program can print the game board to the screen so that the player can see the board.
4. The program can alternate between players so that the program can be played by at least 2 players and up to 10.
5. Player 1 will always go first.
6. The program can be repeated so that the program has the ability to let the players play again.
7. The game board is of size NUM\_ROWS x NUM\_COLUMNS as indicated by user input.
8. Coordinate (0,0) represents the top left corner of the game board.
9. The program has a memory efficient implementation and a fast implementation.

Design

Diagram

Description automatically generatedClass Diagrams

A picture containing table

Description automatically generated

Text

Description automatically generated

Activity Diagrams

GameScreen

Diagram

Description automatically generatedMain()

Diagram

Description automatically generatedgameWon()

Diagram

Description automatically generated

gameDrawn()

Diagram

Description automatically generatedtakeTurn(char player)

Diagram

Description automatically generatedsetBoardFeatures()

IGameBoard (default methods)

Diagram

Description automatically generatedcheckSpace(BoardPosition pos)

checkForWinner(BoardPosition lastPos)

Diagram

Description automatically generated

checkForDraw()

Diagram

Description automatically generated

checkHorizontalWin(BoardPosition lastPos, char player)

Diagram

Description automatically generated

checkVerticallWin(BoardPosition lastPos, char player)

Diagram

Description automatically generated

checkDiagonalWin(BoardPosition lastPos, char player)

Diagram

Description automatically generated

Diagram

Description automatically generated

isPlayerAtPos(BoardPosition pos, char player)

Diagram

Description automatically generated

GameBoard – Primary Implementations

Diagram

Description automatically generatedGameBoard(int r, int c, int numWin) (constructor)

placeMarker(BoardPosition marker, char player)

Diagram

Description automatically generated

whatsAtPos(BoardPosition pos)

Diagram

Description automatically generated

getNumRows()

Diagram

Description automatically generated

Diagram

Description automatically generatedgetNumColumns()

getNumToWin()

Diagram

Description automatically generated

GameBoardMem – Primary Implementations

GameBoardMem(int r, int c, int numWin) (constructor)

Diagram

Description automatically generated

placeMarker(BoardPosition marker, char player)

Diagram

Description automatically generated

whatsAtPos(BoardPosition pos)

Diagram

Description automatically generated

Override of isPlayerAtPos(BoardPosition pos, char player)

Diagram

Description automatically generated

getNumRows()

Diagram

Description automatically generated

Diagram

Description automatically generatedgetNumColumns()

getNumToWin()

Diagram

Description automatically generated

AbsGameBoard

Diagram

Description automatically generatedtoString()

Deployment

Running the extended Tic-Tac-Toe is based off the makefile. To use the makefile navigate to the project directory in the terminal window and type the following commands:

* To compile the code type make
* To run the program and play the game type make run
* To remove the compiled (.class) files type make clean