**GitHub Stuff\project-4---tic-tac-toe-WildestPantaloons\TicTacToeGame.java**

```java
import java.awt.Point;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Everett Wilcox
 *
 *  TicTacToeGame implements TicTacToe to create a game of TicTacToe.
 */
public class TicTacToeGame implements TicTacToe {
    private GameState gameState;
    private BoardChoice[][] gameGrid;
    private List<Point> moves;
    private BoardChoice lastPlayer;

    /**
     * Construct the TicTacToe game.
     */
    public TicTacToeGame() {
        this.gameState = GameState.IN_PROGRESS;
        this.gameGrid = new BoardChoice[3][3];
        this.moves = new ArrayList<>();
        for (int row = 0; row < gameGrid.length; row++) {
            for (int col = 0; col < gameGrid[row].length; col++) {
                gameGrid[row][col] = TicTacToeGame.BoardChoice.OPEN;
            }
        }
        this.lastPlayer = TicTacToeGame.BoardChoice.OPEN;
    }

    /**
     * Creates a new TicTacToe game.
     */
    @Override
    public void newGame() {
        this.gameGrid = new BoardChoice[3][3];
        this.moves.clear();
        this.gameState = GameState.IN_PROGRESS;
        for (int row = 0; row < gameGrid.length; row++) {
            for (int col = 0; col < gameGrid[row].length; col++) {
                gameGrid[row][col] = TicTacToeGame.BoardChoice.OPEN;
            }
        }
        this.lastPlayer = TicTacToeGame.BoardChoice.OPEN;
    }

    /**
     * Checks if the game is Over
     * @return true if the game is over, else false
     */
    @Override
    public boolean gameOver() {
        return this.gameState != GameState.IN_PROGRESS;
```

```
54      }
55
56      /**
57       *  Checks if the player has won the game while also allowing them to pick a move.
58       * @param player the player to check
59       * @return true if the player has won the game, else false
60       */
61      @Override
62      public boolean choose(TicTacToe.BoardChoice player, int row, int col) {
63          if (gameState == GameState.IN_PROGRESS && gameGrid[row][col] ==
TicTacToeGame.BoardChoice.OPEN && player != lastPlayer) {
64              gameGrid[row][col] = player;
65              lastPlayer = player;
66              moves.add(new Point(row, col));
67
68              boolean isOver = false;
69              for (int i = 0; i < 3; i++) {
70
71                  // Check rows for a win
72                  if (gameGrid[i][0] != TicTacToeGame.BoardChoice.OPEN && gameGrid[i][0] ==
gameGrid[i][1]
73                          && gameGrid[i][0] == gameGrid[i][2]) {
74                      isOver = true;
75                  }
76
77                  // Check columns for a win
78                  if (gameGrid[0][i] != TicTacToeGame.BoardChoice.OPEN && gameGrid[0][i] ==
gameGrid[1][i] && gameGrid[0][i] == gameGrid[2][i]) {
79                      isOver = true;
80                  }
81              }
82
83              // Check diagonals for a win
84              if (gameGrid[0][0] != TicTacToeGame.BoardChoice.OPEN && gameGrid[1][1] ==
gameGrid[0][0] && gameGrid[2][2] == gameGrid[0][0]) {
85                  isOver = true;
86              }
87              if (gameGrid[0][2] != TicTacToeGame.BoardChoice.OPEN && gameGrid[1][1] ==
gameGrid[0][2] && gameGrid[2][0] == gameGrid[0][2]) {
88                  isOver = true;
89              }
90
91              // Check who won
92              if (isOver) {
93                  gameState = (player == TicTacToeGame.BoardChoice.X) ? GameState.X_WON :
GameState.O_WON;
94              }
95              // Check for a draw
96              else if (this.moves.size() == 9) {
97                  gameState = GameState.TIE;
98              }
99              return true;
100         }
101         return false;
102     }
103
104     /**
```

```
105         * Get the current state of the game.
106         * @return the current state of the game
107         */
108        @Override
109        public TicTacToe.GameState getGameState() {
110            return this.gameState;
111        }
112
113        /**
114         * Get the game grid.
115         * @return the game grid
116         */
117        @Override
118        public TicTacToe.BoardChoice[][] getGameGrid() {
119            BoardChoice[][] gameGridCopy = new BoardChoice[3][3];
120            for (int row = 0; row < gameGrid.length; row++) {
121                for (int col = 0; col < gameGrid[row].length; col++) {
122                    gameGridCopy[row][col] = gameGrid[row][col];
123                }
124            }
125            return gameGridCopy.clone();
126        }
127
128        /**
129         * Get the list of moves that have been made.
130         * @return the list of moves that have been made
131         */
132        @Override
133        public Point[] getMoves() {
134            return moves.toArray(new Point[0]);
135        }
136
137    }
138
```