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

```java
import java.awt.Point;

/**
 * Interface defining the TicTacToe game model.
 *
 * @author mvail
 */
public interface TicTacToe {
    public static enum BoardChoice {X, O, OPEN};
    public static enum GameState {X_WON, O_WON, TIE, IN_PROGRESS};

    /**
     * Reset the game.
     * All board positions are OPEN, no moves have been made, and the game
     *  is IN_PROGRESS.
     */
    public void newGame();

    /**
     * A choice is invalid if the game is over, the position is
     * out of bounds, the position is already claimed, or the
     * player made the previous choice (no player can make two
     * moves in a row).
     * If the move is valid, claim it for the player.
     * A winning move or choosing the last open position ends
     * the game.
     *
     * @param player expecting either BoardChoice.X or BoardChoice.O
     * @param row row to claim - value from 0 to 2
     * @param col column to claim - value from 0 to 2
     * @return true if the choice was a valid move, else false
     */
    public boolean choose(BoardChoice player, int row, int col);

    /**
     * Return true if either player X or O has achieved
     * 3-in-a-row, whether vertically, horizontally, or diagonally,
     * or if all positions have been claimed without a winner.
     *
     * @return true if the game is over, else false
     */
    public boolean gameOver();

    /**
     * Return the winner (X_WON, O_WON, or TIE) if the game is over,
     * or IN_PROGRESS if the game is not over.
     *
     * @return the winner of a completed game or IN_PROGRESS
     */
    public GameState getGameState();

    /**
     * Get the current game board with each position marked as
```

```
54       * belonging to X, O, or OPEN.
55       * Preserve encapsulation by returning a copy of the original data.
56       *
57       * @return array showing the current game board
58       */
59      public TicTacToe.BoardChoice[][] getGameGrid();
60
61      /**
62       * Get the sequence of moves, where even indexes correspond to the
63       * first player's moves and odd indexes correspond to the second
64       * player's moves.
65       * NOTE: Move rows are stored in the first coordinate, "x", and move
66       * columns are stored in the second coordinate, "y". While possibly
67       * counter-intuitive, it is intentional.
68       * Preserve encapsulation by returning a copy of the original data.
69       *
70       * @return array containing only the sequence of claimed positions
71       *  - i.e. the size of the returned array will match the number of moves
72       */
73      public Point[] getMoves();
74
75  }
76
```