```python
# TTT_state.py - Handles Tic Tac Toe Game State and Moves
# Ari Cohen

from TTT_interface import *
#from TTT_player import *

class Piece():
    X = 3
    O = 4

class GameStatus():
    PLAYING = 1
    QUITTING = 2
    TIE = 3
    WIN = 4
    LOSS = 5

class MoveStatus():
    LEGAL_MOVE = 1
    ILLEGAL_MOVE = 2

# A move consists of two parts
# 1) placing the current piece into an empty square on the board
# 2) selecting a new piece to be the current piece to place
class GameMove():
    LINES = [[0,1,2],[3,4,5],[6,7,8],
             [0,3,6],[1,4,7],[2,5,8],
             [0,4,8],[2,4,6]]

    def __init__(self):
        self.row_placement = -1
        self.col_placement = -1
        self.piece = -1

    def get_row_placement(self):
        return self.row_placement

    def set_row_placement(self, row):
        self.row_placement = row

    def get_col_placement(self):
        return self.col_placement

    def set_col_placement(self, col):
        self.col_placment = col

    def get_piece(self):
        return self.piece

    def set_piece(self, new_piece):
        self.piece = new_piece

    def set_move(self, row, col, new_piece):
        self.row_placement = row
        self.col_placement = col
        self.piece = new_piece
```

```
class GameState():
    AVAILABLE = 1
    UNAVAILABLE = 0
    EMPTY = -1

    def __init__(self, interface_state):
        self.squares = [GameState.EMPTY]*9
        self.interface_state = interface_state
        self.current_player = 1

    def reset(self):
        self.squares = [GameState.EMPTY]*9

    def get_squares(self):
        return self.squares

    def get_square_piece(self, row, col):
        return self.squares[3*row + col]

    def set_square_piece(self, row, col, new_piece):
        self.squares[3*row + col] = new_piece

    def make_move(self, move):
        self.set_square_piece(move.get_row_placement(),
                              move.get_col_placement(),
                              move.get_piece())

    def get_interface(self):
        return self.interface_state

    def set_current_player(self, player):
        self.current_player = player

    def get_current_player(self):
        return self.current_player

    def toggle_players(self):
        # Oscillate between 1 and 2
        self.current_player = ((self.current_player * 2) % 3)

    def get_current_piece(self):
        if (self.current_player == 1):
            return Piece.X
        else:
            return Piece.O


def check_pieces_for_win(p1, p2, p3):
    return (p1 == p2 == p3 != GameState.EMPTY)


def check_for_win(game_state):
    squares = game_state.get_squares()
    for places in GameMove.LINES:
        if check_pieces_for_win(squares[places[0]], squares[places[1]],
                                squares[places[2]]):
            return GameStatus.WIN
    # no win, so check for tie
    tie = True
    for square in squares:
        if (square == GameState.EMPTY):
            tie = False
    if (tie):
        return GameStatus.TIE
    else:
        return GameStatus.PLAYING

def copy_game_state(game_state):
    new_game_state = GameState(game_state.interface_state)
    new_game_state.set_current_player(game_state.get_current_player())
    square_list = game_state.get_squares()
    for piece in range(9):
        if(square_list[piece] != GameState.EMPTY):
            new_game_state.set_square_piece(piece/3,piece%3,square_list[piece])
    return new_game_state
```