Project 11

This assignment is worth 55 points (5.5% of the course grade) and must be **completed and turned in before 11:59 on Tuesday, December 5, 2017.**

Assignment Overview

- Classes

Assignment Background

You will implement a game called **Gomoku** in Python using classes. Gomoku is an abstract strategy board game. It is traditionally played with Go pieces (black and white stones) on a Go board with 15x15 intersections.

How to play: The black plays first if white did not win in the previous game, and players alternate in placing a stone of their color on an empty intersection. The winner is the first player to get an unbroken row of five stones horizontally, vertically, or diagonally. You can play with the game before starting this project to help you understand this project: http://gomokuonline.com/

For Testing we will test with different board dimensions and different number of pieces in a row for a win — although traditionally 15x15 with 5 in a row there are many different variations played around the world.

The `MyError` exception is provided so exceptions can be called both in the main program and in the classes.

Project Specifications

You must implement the following classes and methods. *You are not allowed to add methods and you are only allowed to access attributes using the provided methods (that is no "name mangling" allowed).*

1. `class GoPiece`. It represents black or white pieces used in the game. You need to implement the following methods & attributes.:
    a. `def __init__(self, color)`: this method creates a Gomoku piece. It has an attribute named 'color' which is either `'black'` or `'white'`. The attribute must be private, i.e. the name is preceded by double underscores. Raise `MyError('Wrong color.')` for a color that is not black or white. Set the default value to `'black'`
    b. `def __str__(self)`: this method displays the black piece as ' ● ', and white piece as 'o' .
    c. `def get_color(self)`: this method returns the color of the piece as a string `'black'` or `'white'`.

2. `class Gomoku.` It contains methods for setting up the game, displaying the game board, and playing the game. You need to implement the following methods & attributes as specified. All attributes must be private:
    a. Method `__init__(self,board_size,win_count,current_player)` with four attributes, three have defaults, all must be private.
        i. `board_size` : the size of the game board, which by default is 15 (representing a 15x15 board). Check that the value is an int so it will raise a `ValueError` if not.
        ii. `win_count` : the number of pieces in an unbroken row that a player must get to win, which by default is 5. Check that the value is an int so it will raise a `ValueError` if not.
        iii. `current_player` : the color of the current player. Default is `'black'`. Raise `MyError('Wrong color.')` for color that is not black or white.
        iv. `go_board`: a list of lists to represent the game board. Specifically, it is a list of row-lists.
    b. Method `assign_piece(self, piece, row, column)`: it places the piece (a `GoPiece` object) at the specified position on the game board. Row 1 is the top row, column 1 is the left-most column. Raise `MyError('Invalid position.')` if the specified row or column is too big or too small to be on the game board. Raise `MyError('Position is occupied.')` if the space is already occupied by a piece.
    c. Method `get_current_player(self)`: it returns the current player as a string `'black'`or `'white'`.
    d. Method `switch_current_player(self)`: it returns the 'other' player as a string, that is, if the `current_player` is `'white'`it returns the string `'black'`otherwise it returns `'white'`.
    e. Method `__str__(self)`: returns a string that represents the board for printing. We provide the complete method.
    f. Method `current_player_is_winner()`: This method returns `True` when `current_player` is a winner, otherwise `False`. It will check all rows and columns to check if there is an unbroken sequence of size `win_count` (default is five-in-a-row). Important: first implement this to test for horizontal and vertical winners – leave the diagonal test until everything else in the project is correct (only one test considers diagonals—see test descriptions).
3. `main()` function: it creates an instance of the class `Gomoku`, It will alternatively let the black and white players place their piece on the game board, until there is a winner, or until a player inputs 'q' to quit. Each time a player places a piece on the game board, this function will check whether a winner is generated using the method `current_player_is_winner`, and display the current game board. Remember to

check that the row and column are ints. If anywhere in the program a `ValueError` is raised, catch it in the main function.

Deliverables

The deliverable for this assignment is the following file:

`proj11.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via the **Mimir system** before the project deadline.

Sample Output

Class Test `GoPiece` `tests the GoPiece __init__ and __str__ methods.`

Class Test `Gomoku` `tests the Gomoku __init__ and __str__ methods`

Method Test `get_current_player` `tests get_current_player, switch_current_player and tests some defaults to __init__.`

Method Test 1 `current_player_winner` `tests losing and winning boards, but only for horizontal and vertical wins (no diagonals tested).`

Method Test 2 `current_player_winner` `tests losing and winning boards, but only for diagonal wins`

Error Test `tests errors raised in classes`

`Test 1` `input1.txt and output1.txt are on web page`

`Test 2` `includes tests of a number of errors`

`input2.txt and output2.txt are on web page`

`Test 3 Blind Test`

Grading Rubric

```
General Requirements:

  (4 pts) Coding Standard 1-9

Implementation:
```

(5 pts) GoPiece

(4 pts) Private attributes in classes (no Mimir test)

(4 pts) Gomoku __init__

(5 pts) Gomoku assign_piece

(5 pts) Gomoku get_current_player and switch_current_player

(5 pts) Gomoku current_player_is_winner (horizontal & vertical)

(5 pts) Gomoku current_player_is_winner (diagonal)

(5 pts) Test1

(4 pts) Test2

(4 pts) Test3: blind test

(5 pts) Specified errors are checked and handled