CS 245 - Lab 2 Fall, 2021

Lab 2 - Tic Tac Toe Winner

The goal of this assignment is to demonstrate your mastery of matrices(2D arrays). Download your code here.

Background

According to Wikipedia, Tic Tac Toe is a “paper-and-pencil” game for two players. It is also known as the Xs and Os game because it requires two players to place three of their marks(either X or O) in a diagonal, horizontal, or vertical row. That is how you win a tic tac toe game. Each player takes turns to mark the spaces in a 3x3 grid. An example of the tic tac toe game is shown below.

```
    0   1   2
  -----------
0 | X | O | X |
  -----------
1 | X | O | O |
  -----------
2 | O | X | X |
  -----------
   3x3 grid
```

Figure 1: An example of a completely full Tic Tac Toe game board
The game has no winner for this round.

Here are some rules for this Tic Tac Toe assignment:

- Players take turns placing characters into empty squares (" ").
- The first player *A* always places "X" characters, while the second player *B* always places "O" characters.
- "X" and "O" characters are always placed into empty squares, never on filled ones.
- The game ends when there are 3 of the same (non-empty) character filling any row, column, or diagonal.
- The game also ends if all squares are non-empty.
- No more moves can be played if the game is over.

Given an 2D array ***moves*** that record all the moves of player A and B, where each element in the array is corresponding to the row and column of the grid that the player marks their own character(“X” or “O”) in the order that A always plays first.

Return the winner of the game if it exists (***A*** or ***B***), in case the game ends in a draw return "**Draw**", if there are still movements to play return "**Pending**".

You can assume that all elements in the array ***moves*** are **valid**(follow the rules of tic tac toe), the grid is originally empty and player A will place the mark first.

Examples:

```
Input: moves = [[0,0],[2,0],[1,1],[2,1],[2,2]]

Output: "A"
```

```
 -----------
| X |   |   |
 -----------
|   | X |   |
 -----------
| O | O | X |
 -----------
```

```
Figure 1: Player A won the game
```

```
Input: moves = [[0,0],[1,1],[0,1],[0,2],[1,0],[2,0]]

Output: "B"
```

```
 -----------
| X | X | O |
 -----------
| X | O |   |
 -----------
| O |   |   |
 -----------
```

```
Figure 2: Player B won the game
```

```
Input: moves = [[0,0],[1,1],[0,2],[0,1],[2,2],[1,2],[2,1],[2,0],[1,0]]

Output: "Draw"
```

```
 -----------
| X | O | X |
 -----------
| X | O | O |
 -----------
| O | X | X |
 -----------
```

```
Figure 3: The game ends in draw and there are no moves to take
```

```
Input: moves = [[0,0],[0,1],[2,1]]

Output: "Pending"
```

```
 -----------
| X | O |   |
 -----------
|   |   |   |
 -----------
|   | X |   |
 -----------
```

`Figure 4: The game results in pending since it is not finished yet`

Requirements

- You must submit a single Java file containing the class named **TicTacToeWinner**. You class must contain the function named “`ttcWinner`” with the following header:

  ```
  public String ttcWinner(int[][] moves)
  ```

- The function “`ttcWinner`” **must return** a string value(“A”, “B”, “Draw”, or “Pending”), and your class may contain a main function for testing, along with other helper functions.

- Your function should be able to pass all the test cases, examples listed above aren’t the grading scale of your implementation.

Recommendations

Recommendation 1: You can use the “%” operator to determine whether the move in the array belongs to player **A** or player **B**(A always places the mark first). Then you can store the move into your own Tic-tac-toe board(either represented by “X” and “O”, or integers, your choice).

Recommendation 2: You can have helper functions, for example, a boolean function named “`checkWinner`”. The purpose of this function is to check whether a player has successfully placed their marks horizontally, vertically, diagonally, or anti-diagonally in the 3x3 board.

Submission

- Download the source code for your implementation from Github Classroom. If you don’t have a Github account, create one via github.com > Sign up.
- Select your name from the list of names available, which will link your GitHub ID. If you do not see your name on the list, speak with the instructor or one of the teaching assistants.
- Submit the repository link on Canvas. You may also add any comments in a README file (text, PDF or Word document) to help the grader understand or execute your implementation.

Grading

Your grade for this assignment will be determined as follows:

- 75% = Implementation: your class implementations must run successfully with the source files and data provided. It must produce the expected results, a sample of which appears in the Implementation section above. Any deviation from the expected results results in 0 credit for implementation.
- 15% = Decomposition: in the eyes of the grader, the degree to which your solution follows the suggestions above or otherwise represents a reasonable object-oriented and procedural decomposition to this problem.
- 10% = Style: your code must be readable to the point of being self-documenting; in other words, it must have consistent comments describing the purpose of each class and the purpose of each function within a class. Names for variables and functions must be descriptive, and any code which is not straightforward or is in any way difficult to understand must be described with comments.