Exercise 2 – Reverse Tic-Tac-Toe for Console

Objective

- Using Classes for implementing Object Oriented Programing concepts
- Using Constructors, Enums, Properties, Access modifiers, Modifiers
- Working with Arrays / Collections / Data Structures
- Use of the String Class
- Referencing External DII (Assembly)

Prior Knowledge

- Acquaintance with Microsoft Visual Studio .NET
- Basic C# Syntax
- Working with Arrays / Collections / Data Structures
- Working with Classes (Access modifiers, Constructors, Properties)
- Use of the String Class
- Referencing External Libraries (Assembly)

Pre – Preparing

- Microsoft Visual Studio installed
- Ex02.ConsoleUtils.dll file (part of the download package)

The Exercise

Implementing the **Reverse** **Tic-Tac-Toe** game for Console

The Program:

The program will allow two human players to play against eachother in turns, or for a human player to play against the computer.

The program will display an empty board of of cells:

```
   1   2   3   4   5
1|   |   |   |   |   |
 ====================
2|   |   |   |   |   |
 ====================
3|   |   |   |   |   |
 ====================
4|   |   |   |   |   |
 ====================
5|   |   |   |   |   |
 ====================
```

Each turn, the user will be asked to choose in which cell he wants to put his ‘coin’ (‘X’ or ‘O’).

The goal is to **avoid** having a sequence of N coins (N is the width of the board).

The flow:

1. The user will be asked to choose board size (minimum 3x3, maximum 9x9). The board must be a square (rows == cols).
2. The user will be asked to choose weather to play against a computer upponent or another human upponent.
3. The game starts with an initialize board (5x5 example):

```
   1   2   3   4   5
1|   |   |   |   |   |
 ====================
2|   |   |   |   |   |
 ====================
3|   |   |   |   |   |
 ====================
4|   |   |   |   |   |
 ====================
5|   |   |   |   |   |
 ====================

Dani's turn:
```

4. Each turn, the user will be asked to choose in which cell he wants to put his ‘coin’. If the cell is 'illegal' (occupied or out of bounds), the user will be prompted to enter a valid choice, and so on until he chooses a legal choice (a legal choice is an empty cell in the board. Any other input is invalid and a proper message should be displayed to the user).
5. After a valid input was entered by the user, the screen will be cleared and the board will be re-drawn with the selected cell 'exposed':

```
  1   2   3   4   5
1|   |   |   |   |   |
 ====================
2| X |   |   |   |   |
 ====================
3|   |   | O | O |   |
 ====================
4|   | X |   | X |   |
 ====================
5|   |   | X |   | O |
 ====================
```

6. If the new move did not complete a sequence of N coins (5 coins in this case), the turn will move to the opponent (back to phase 5).
7. If the new move DID complete a sequence of N coins, this is the losing player. The winner (the one without a sequence of N coins) will get a point and the current updated score will be displayed.
8. If the new mode did not complete a sequence and the board is full, a TIE message will be displayed.
9. After a tie or a win, The user will be ask to decide if he wants to play another round, or to quit.
10. Starting stage 5, the user can quit the app at any point by typing 'Q' as input.

General Instructions

- You may use the Next method of class Random for randomization (to produce computer-based 'choices')
- You must validate each input and prompt the user in case of any invalid input. You must differentiate between **syntax-invalid** inputs and **logical-invalid** inputs:
  - Syntax-invalid: inputing a number instead of a letter
  - Logical-invalid: inputing an out of range letter (i.e. Y)
- **Before printing each stage of the game board, you must clear the screen. For this, use the file Ex02.ConsoleUtils.dll, which contains the service class Ex02.ConsoleUtilts.Screen and use the static method Clear().**
  To use this dll:
  - Right click in the Solution Explorer window, above the project's Refrences.
  - Choose 'Add Reference' and then choose the dll using the 'Browse' option.
  - Now, you are able to access the Ex02.ConsoleUtils class, and use the methods of that class as you'd use any other normal library methods.
  Important – Do not attach this file in your submission – GMAIL will reject your work.
- **Architecture and Software Engineering:**
  - Use Object Oriented architecture!
  - Demonstrate correct use of C# 2.0 and .Net capabilities
  - **Apply correct separation and segregation between the Classes who manage the logics and data of the game and the Classes who are responsible for the UI and User Interactions.**
    In other words: Seperation between the implementation of the User Interface (UI) and the implementation of the system's logic.
    You must keep in mind that these implementations will serve you when you will want to develop the game with an alternative UI in the future, and the aspiration is to have as many reusable parts (unchanged) as possible.
- Implemeting AI-based computer 'choices' (instead of just random choices) will gain up to 7 bonus points.
- You may use in the course's facebook group in order to ask questions regarding this assignemt.
- You must comply with the coding standards, as stated in the relevant document, found on the course website. **Pay extra attention to the standards of Class Field Names and Method Parameters**. Points will be deducted to whom ever does not comply with these standards.
- Send your submission to the email address as described in the "Ex_Submit_Instructions_DN_IDC_21B.pdf" document, which can be found in the course's site. Points will be deducted for not following the instructions carefully.
- Avoid cheating (Do not use other students' assignments as a basis for yours. Refrain from copying the work of fellow students from your group or previous semesters. Cheaters will be caught and punished. Work independently!)
- Submission is due to May 16th 2021, 22:00.

Good Luck ☺