# Exercise 2 – American Checkers 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 knowledge
- Working with Arrays / Collections / Data Structures
- Work with Classes (Access modifiers, Constructors, Properties)
- Use of the String Class
- Referencing External DII (Assembly)

## Pre – Preparing

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

## The Exercise

You must implement the game American Checkers for Console.

Here are the rules of the game.

**The Program:**

The program will allow two human players to play against eachother by turns, or for a human player to play against the computer which randomly chooses a move out of the current set of legal moves.

**A bonus of up to 8 points** will be given for AI implementation of the computer's moves (In case of AI implementation, please write a short explanation about your AI implementation in the comments part in the body of the submission email).

**The flow:**

1. The user will be asked to enter his name (no spaces, max 10 chars).
2. The user will be asked to choose board size (6, 8 or 10)
3. The user will be asked to choose weather to play against a computer upponent or another human upponent. In case of another human uponent, the user will be asked to enter the upponents name (no spaces, max 10 chars).
4. The game starts with an initialize board (8x8 example):

```
   A   B   C   D   E   F   G   H
 =================================
a|   | O |   | O |   | O |   | O |
 =================================
b| O |   | O |   | O |   | O |   |
 =================================
c|   | O |   | O |   | O |   | O |
 =================================
d|   |   |   |   |   |   |   |   |
 =================================
e|   |   |   |   |   |   |   |   |
 =================================
f| X |   | X |   | X |   | X |   |
 =================================
g|   | X |   | X |   | X |   | X |
 =================================
h| X |   | X |   | X |   | X |   |
 =================================
Dani's turn (x):
```

The setup will always include 2 empty rows between the upponents. Meaning that in a 10x10 board, there will be 4 rows of coins per player, and in a 6x6 board, there will be 2 rows of coins per player.

5. At each stage, the player will be asked to enter his 'move' by this format: COLrow>COLrow (i.e. Af>Be)
6. If the move is 'illegal', the user will be prompted to enter a valid move.
7. After a valid move was entered by the user, the screen will be cleared and the board will be re-drawn with the new state:

```
   A   B   C   D   E   F   G   H
 =================================
a|   | O |   | O |   | O |   | O |
 =================================
b| O |   | O |   | O |   | O |   |
 =================================
c|   | O |   | O |   | O |   | O |
 =================================
d|   |   |   |   |   |   |   |   |
 =================================
e|   |   |   |   |   |   |   | X |
 =================================
f| X |   | X |   | X |   |   |   |
 =================================
g|   | X |   | X |   | X |   | X |
 =================================
h| X |   | X |   | X |   | X |   |
 =================================
Dan's move was (X): Gf>He
Avi's Turn (O):
```

8. In case of a 'jump' over an opposing coin ("Eating" / "Capturing"), the opposing man is eliminated. In order to allow a sequence of jumps, after each jump, the program will check if there is another valid jump option for the current coin. If so, the turn is still the current player's, and the following jump(s) will be his only valid move(s).
9. Ths O's kings are marked with 'Q'. The X's kings are marked with Z.
10. If the move did not create a draw or a win, the turn goes to the upponent.
11. In case of a draw, a draw will be prompted.
12. In case of a win the winner will gain points according the number of coins difference between him and the upponent (A king is worth 4 points) and a win will be prompted indlucing the current aggregated score.
13. The user may decide to quit by entering 'Q' instead of a valid move.
14. After a win/draw/quit the user will be asked to decide weather to play another round or to quit the program. Another round will be with the same configuration as the previous one. The players will continue to gain points.
15. In case the upponent is the computer, the flow remains the same, only that the computer does not wait to be prompted to make a move.

Again: The full set of rules and description can be found here.

## General Instructions

- You may use the Next method from the Random Class for randomization.
- You must validate each input and prompt the user in case of an 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 call this dll, place the dll in C:\Temp – (**Important – Do not place it anywhere else!)**
  - Right click in the Solution Explorer window, above the project's Refrences.
  - Pick Add Reference and then choose the dll using Browse.

  Now, you are able to access the Ex02.ConsoleUtils class (which is defined in the dll) 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 (The checker has the file in the same location)
- **Architecture and Software Engineering:**
  - You must use Object Oriented architecture!
  - You must demonstrate correct use of C# and .Net capabilities
  - **You must apply correct 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 for "Windows", and the aspiration is to have as many reusable parts (unchanged) as possible.

- 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 Function Parameters**. Points will be deducted to whom ever does not comply with these standards.
- 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 11th 2022, 22:00.

**Good Luck ☺**