Yatzy Exam Assignment

Introduction

You must create a C# program that can play yatzy with six dice (https://en.wikipedia.org/wiki/Yatzy). The yatzy game is for a one-person game.

You must hand in a running program! If the program is not running you cannot take the exam!

Program Structure

There must be classes for the following in the code.

- A fair dice and a biased dice
- A turn (or a roll) more on this later
- The complete yatzy game

You are free to add additional classes or interfaces and make usage of inheritance where you find it appropriate. You are free to name the classes. However, the readability of the code is important.

In the following, there is a listing of the requirements for each of the program structures listed above. These requirements are split into data requirements, i.e., what should be stored for an object and functionality, i.e., which methods should there be on each of the classes.

Requirements to the Dice Classes

- You must implement a six-sided fair dice
- You must implement a biased six-sided dice

A *positively* biased dice is a dice that on average does better than the fair dice. A *negatively* biased dice is a dice that on average does worse.

There are the following functional requirements for the dice classes.

- You must be able to hold the current value of the dice
- You must be able to change the degree of a positively/negatively biased dice during the game
- You must be able to demonstrate (in code) that a dice is fair/positive-biased/negative biased

Requirements to Turn/Roll

For a player's turn, the player must roll all six dice in the first roll. The player must then be able to hold the "good" dice, e.g., all fivers and sixers and roll the remaining dice.

There are the following data requirements to a turn/roll

- The number of rolls per turn is default 3 must it must be possible to change it to a value between 2 and 10
- You must be able to nicely print the result of the turn, i.e., what do the six dice show when the turn is over. This should be in a textual format (Console.WriteLine)

Requirements to the Yatzy Game

The game must support the following outcomes of a turn. See the web https://en.wikipedia.org/wiki/Yatzy for the definition of these outcomes

- Ones/twos/threes/fours/fives/sixes
- One pair/two pairs/three-of-a-kind/four-of-a-kind
- Small straight/large straight
- Full house
- Chance
- Yatzy

The game is done in two sections

- In the upper section, the player must do ones/twos/threes/fours/fives/sixes in any order
- If the total score of the upper section is 63 points or above the player gets a 50 points bonus
- In the lower section, the remaining outcomes are played in any order

The yatzy game must support the following functionality.

- Play the two sections in order
- Support the bonus
- Automatically list the possible outcomes of the current status of the turn
- Keep track of the outcomes already used by the player and the score for each
- Keep track of the total score
- Be able to drop an outcome (get a zero score), e.g., if the player only has full-house outcome left and does not have full-house

Documentation

You **must** write a very short 0.5-1.0 page document that describes the overall structure of the program. In addition, you must document any assumptions that you have made. If you have use code that is not in the .NET library you must clearly document where the code is from and what you have used it for.

The documentation must be added as a comment at the top of the files that deal with the game class.

Additional Information

There are the following additional overall requirements for the codebase.

- The code **must** follow the Microsoft C# Coding Convention.
- The code **must** be well-structured and highly readable by humans.
- Each class **must** have appropriate constructors.
- Each class **must** have appropriate properties.
- There **must** be an appropriate use of the basic data types such as int and string.
- There **must** be an appropriate use of access modifiers, e.g., private, protected, and public.
- There **must** be an appropriate use of parameters and return values from methods.
- There **must** be at least one method that raises an exception.
- There **must** be an appropriate use of the collection classes this includes the use of generics.

You **can** consider the following additional requirements.

- You **can** add variables, properties, and methods that can assist you in the implementation.
- You **can** use abstract classes and interfaces.
- Parts of the public interface **can** be tested using the C# Unit-Test Framework.
- Parts of the public interface **can** be documented using the C# XML Documentation features.
- You **can** do simple error checking of for example actual parameters. We will overall assume that the user of the program behaves appropriately.

User interface

- It is **cannot** have a graphical user interface.
  - This is to focus on the core object-oriented concepts
- A textual interface must bed used, the code for this interface should be well-structured and readable.

Handing in of Exam Assignment

- Will be announced later