Programming II

Maximum points for this work is 9 marks.

Assignment 1 – Implement a Rational Class - Due in Week 3

Once you are done with your program, upload it to eCentennial under Assessments / Assignment / Assignment 1

A rational number is a number that can be expressed as a ratio of whole numbers. When expressed as a decimal, a rational number has a finite or recurring expansion (Google search). We will use two integers to represent a Rational Object. The object must behave like normal fraction i.e. must do addition and subtraction correctly.

You are not required to reduce the object to its lowest form or to make "proper" rationals. You will be penalized for poor coding practices such as the following:

Plagiarism will result at minimum zero mark, but it may be escalated to more serious punitive measures.

1. Unusual indentation.
2. Erratic empty lines.
3. Non-descriptive identifiers.
4. Write readable, clear and simple code. (So code documentation is not necessary)

The Rational Class

This class consist of six members as shown below:

Rational **Class**
Properties
+ Denominator : int «set private» + Numerator : int «set private»
Methods
+ «constructor» Rational(numerator= 0, denominator =1) + ToString() : string + IncreaseBy(Rational other) : void + DecreaseBy(Rational other) : void

You will implement the Rational Class in Visual Studio. A short description of the class members is given below:

Properties:

You must use auto-implemented properties.

The getters are public, and the setters are private

0.5 mark 1. **`Denominator`** – this auto-implemented property is an integer representing the bottom of the rational number. The getter is public and the setter is private.

0.5 mark 2. **`Numerator`** – this auto-implemented property is an integer representing the top of the rational number. The getter is public and the setter is private.

Constructor:

1 mark 1. **`public Rational(int numerator = 0, int denominator = 1)`** – This is the public constructor. It takes two parameters: integers representing the numerator (default is 0) and the denominator (default is 1). The method assigns the two arguments to the appropriate fields.

Methods:

1 mark 1. **`public override string ToString()`** – This method overrides the same method of the Object class. It does not take any parameter but return a string representation of itself. You decide on the format for the output.

2 marks 2. **`public void IncreaseBy(Rational other)`** – This is a public method that takes an argument of type `Rational`. It adds the argument to the current object. This method does not output anything to the screen, return a value nor does it mutate (change) the argument.

1 marks 3. **`public void DecreaseBy(Rational other)`** – This is a public method that takes an argument of type `Rational`. It subtracts the argument from the current object. This method does not output anything to the screen, return a value nor does it mutate or change the argument.

Testing

In your test harness do the following:

Your output should make sense. You should display the two `Rational` before the operation, inform the user what kind of operation will be done, do the operation and then display the fraction after the operation.

1 mark 1. Create 4 `Rational` objects using **`0`**, **`1`** and **`2`** arguments. Try to use named arguments in the last instantiation.

1 mark 2. Select two pairs of the above object, print them, then call the `IncreasedBy(Rational)` method and print the objects again.

1 mark 3. Select another two pairs of the above objects, print them then call the `DecreasedBy(Rational)` method and print the objects again..

Compose your output so that the user will be able to understand the what is happening.