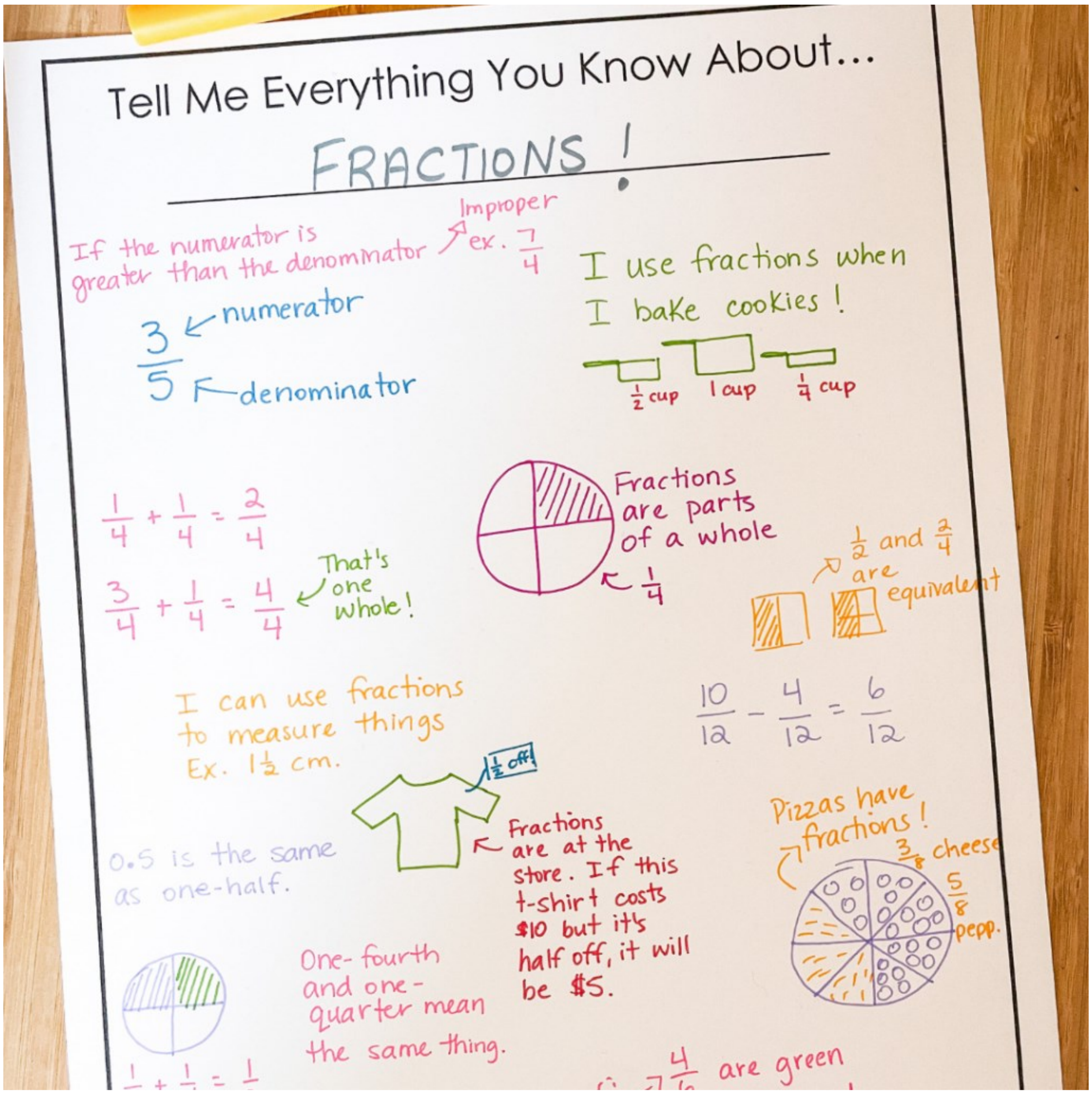
Permanente Evaluatie 3 – Fractions

Opleiding:	Bachelor Elektronica-ICT
Opleidingsfase:	1
Vak:	Object Oriented Programming 1 Docent: Ronny Mees Versie A

A fraction shows part of a whole. This whole can be a region or a collection. The word fraction is derived from the Latin word "fractio" which means 'to break'. The Egyptians, being the earliest civilization to study fractions, used fractions to resolve their mathematical problems, which included the division of food, supplies, and the absence of a bullion currency.

In Ancient Rome, fractions were only written using words to describe a part of the whole. In India, the fractions were first written with one number above another (numerator and denominator), but without a line. It was the Arabs only, who added the line which is used to separate the numerator and the denominator.

A fraction has two parts. The number on the top of the line is called the **numerator**. It tells how many equal parts of the whole or collection are taken. The number below the line is called the **denominator**. It shows the total number of equal parts the whole is divided into or the total number of the same objects in a collection.



Tell Me Everything You Know About...
FRACTIONS!
If the numerator is greater than the denominator
Improper ex. 7/4
3/5 ← numerator ← denominator
I use fractions when I bake cookies!
1/2 cup 1 cup 1/4 cup
1/4 + 1/4 = 2/4
3/4 + 1/4 = 4/4 That's one whole!
Fractions are parts of a whole
1/4
1/2 and 2/4 are equivalent
I can use fractions to measure things Ex. 1½ cm.
10/12 − 4/12 = 6/12
½ off
Fractions are at the store. If this t-shirt costs $10 but it's half off, it will be $5.
Pizzas have fractions! 3/8 cheese 5/8 pepp.
0.5 is the same as one-half.
One-fourth and one-quarter mean the same thing.
4 are green

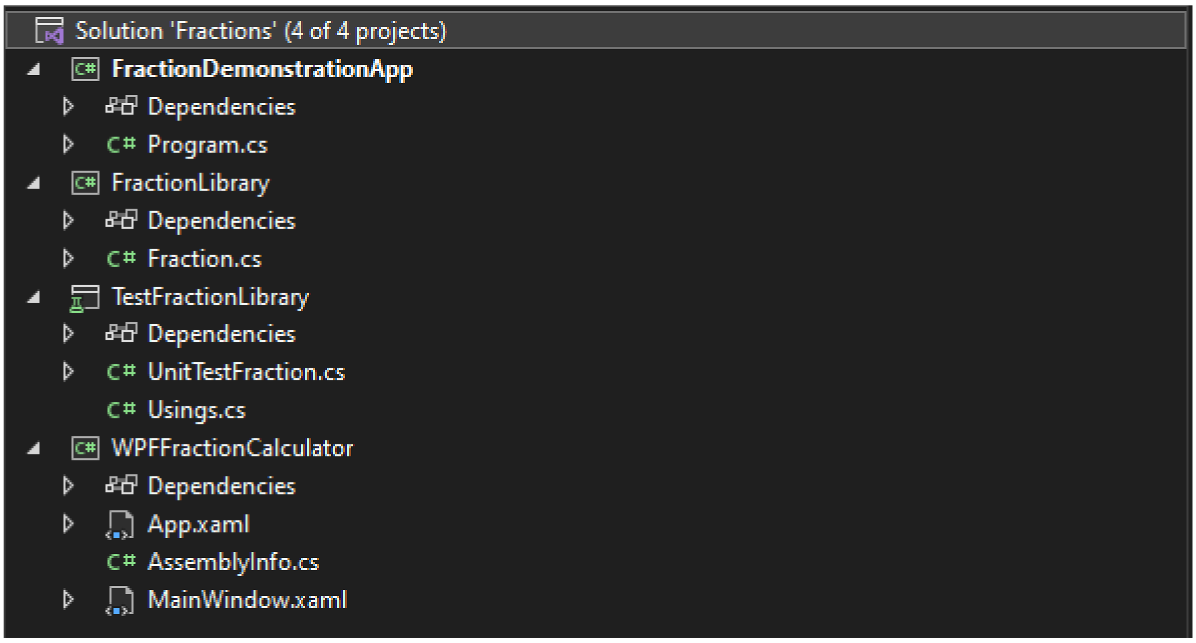
Source: https://shelleygrayteaching.com/fun-with-fractions-how-to-help-your-students-love-learning-about-fractions/

The goal of this challenge is to create a Fraction class as part of a library, which can be used to solve fraction computations in both console and graphical applications. The library will need to be thoroughly tested, that is where the unit tests come into play.

So in this challenge you will need to provide the following **projects in a single solution**:

- A class library containing the Fraction class
- Unit tests for this Fraction class
- A console application to demonstrate the basic usage of the class
- A graphical WPF application to allow the user to solve fractional computations

So basically you should end up with a similar solution as shown in the image below:



```
Solution 'Fractions' (4 of 4 projects)
  FractionDemonstrationApp
    Dependencies
    Program.cs
  FractionLibrary
    Dependencies
    Fraction.cs
  TestFractionLibrary
    Dependencies
    UnitTestFraction.cs
    Usings.cs
  WPFFractionCalculator
    Dependencies
    App.xaml
    AssemblyInfo.cs
    MainWindow.xaml
```

Each of these are described in more detail in the following sections.

THE FRACTION CONSOLE APPLICATION

It is probable best to start by creating a new Console Application within a solution and add projects from there on.

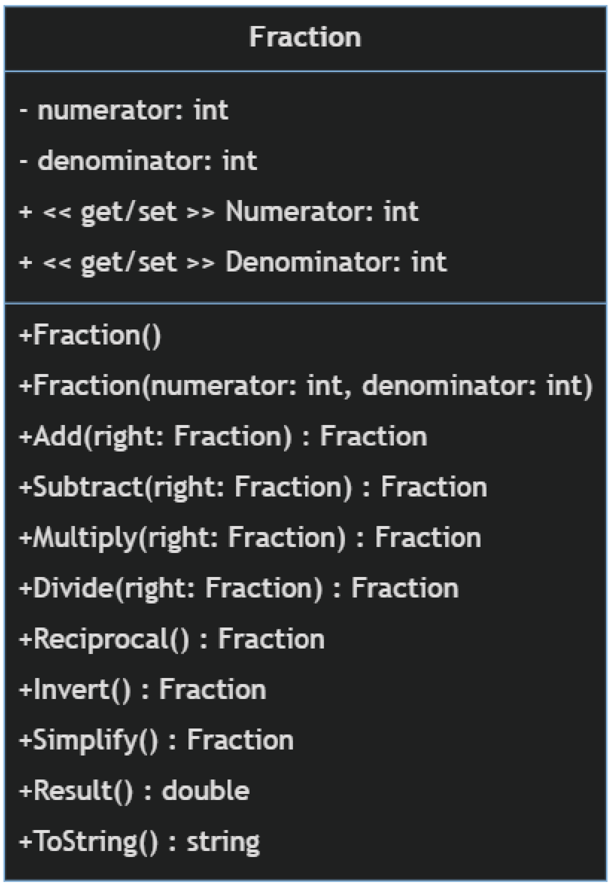
This application serves as a demonstrator of the Fraction library and should provide some code snippets that show how to use the library.

A partial example:

```
Fraction oneFourth = new Fraction(1, 4);
Fraction twoFourth = new Fraction(2, 4);
Fraction sum = oneFourth.Add(twoFourth);
// This should output: 3 / 4
Console.WriteLine(sum);
Console.WriteLine($"Which basically is the same as {sum.Result()}");
```

THE FRACTION LIBRARY

Create a library as described Chapter 42 - Creating Libraries. Add a class called Fraction and implement the class so it adheres to the following UML diagram.


Fraction
- numerator: int
- denominator: int
+ << get/set >> Numerator: int
+ << get/set >> Denominator: int
+Fraction()
+Fraction(numerator: int, denominator: int)
+Add(right: Fraction) : Fraction
+Subtract(right: Fraction) : Fraction
+Multiply(right: Fraction) : Fraction
+Divide(right: Fraction) : Fraction
+Reciprocal() : Fraction
+Invert() : Fraction
+Simplify() : Fraction
+Result() : double
+ToString() : string

The general idea behind the mathematical methods is that the result of the calculation is always returned as a new object of type Fraction. Neither the called upon object or the right operand is ever modified.

Most methods are self-explanatory. The rest are explained briefly here:

- Reciprocal() returns a Fraction where the numerator and denominator are swapped.
- Invert() returns a Fraction where the numerator has the opposite sign
- Simplify() returns a Fraction where the signs of both enumerator and denominator are cleaned up. It also contains the smallest numerator and denominator values that still match the original Fraction.
  - Sign simplification results for example in a positive sign if both the numerator and denominator are negative. If either is negative and the other positive, the simplified version should have a negative numerator.
  - For the fraction simplification itself, you will need to determine the greatest common factor (GCF) and divide both the numerator and denominator by this factor.
- Result() returns the actual floating point approximation of the Fraction.
- ToString() returns a clean string representation of the Fraction. For example: -3 / 4.

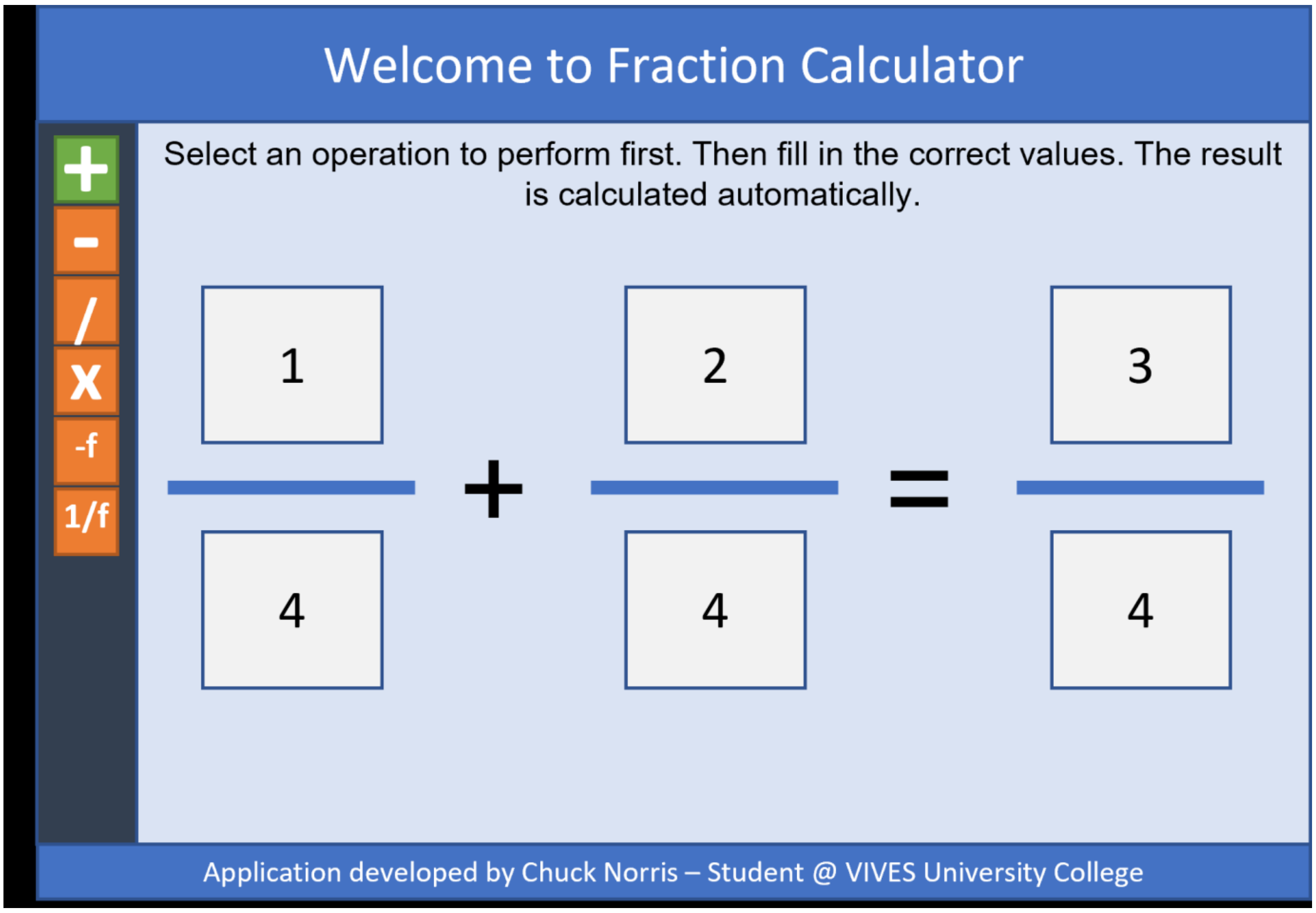
The denominator should never have a value of 0. Dividing by zero will throw an exception and lead to an application crash. To avoid this, make sure to only set the new denominator value in the setter if the incoming value is different from 0.

Fraction Unit Tests

Add an xUnit project to the solution as described in Chapter 43 - Unit Testing.
Try to follow the Test Driven Development methodology as much as possible when implementing the Fraction class.
Make sure to create at least 1 test method for each method of the Fraction class.

Fraction Calculator - A Graphical WPF Application

Create a WPF application that serves as a fraction calculator for the user. The image below shows a mockup of all the functionality that should be available.


Welcome to Fraction Calculator
+
-
/
X
-f
1/f
Select an operation to perform first. Then fill in the correct values. The result is calculated automatically.
1
4
+
2
4
=
3
4
Application developed by Chuck Norris – Student @ VIVES University College

Do not try to copy this exact interface. Make it your own ! You can add more information, images, controls, ...

Some things to keep in mind:

- Create your own style but keep it user-friendly and clean
- All operations should be available
- The user should be warned if a numerator of 0 is provided
- Selecting the operations Invert() and Reciprocal() should disable/hide one of the operands input fields.
- Provide an About window that shows some basic information about the application (author, year, description, link to github, ...). Allow the user to access this window.

THE README

Last but not least you will also need to provide a README.md file in your repository with the following sections:

- Project description: a short description of the whole solution and the different projects in the solution
- Author: Your name
- Screenshots: a couple of screenshots of the application
- Setup and Usage: short description of what applications and tools are required to use the app (Visual Studio, ...)
- Unit Tests: Explain what is being unit tested and how to run the unit tests
- UML Class diagram of Fraction
- Future Improvements: explain the things that do not work or what could be done better