## Why a Class?

Imagine that the company you work for is going to create a series of lessons on fractions. It might be useful to create a Fraction class that all of the fraction lessons could use.

A fraction has a numerator and denominator. For instance 1/2 has a numerator of 1 and a denominator of 2.

The string version of a fraction is the numerator, "/", and the denominator.

We can convert a fraction to decimal by dividing the numerator by the denominator (making sure that the denominator is not zero, of course!)

An improper fraction is one where the numerator is equal or greater than the denominator: 4/3 for instance.

We can create a Fraction class with all of the features above incorporated. This Fraction class can then be used by anyone who write one of the lessons on Fractions.

## The Fraction Class

Start a new Visual Basic program called **TestFraction**. We will thoroughly test the class before we let other programmers use it.

From the menu select Project-> Add Class. Name the class **Fraction**.

Write the code for the class as shown below:

```
Public Class Fraction
    Private m_numerator As Integer = 0
    Private m_denominator As Integer = 1

    Public Property Numerator() As Integer
        Get
            Return m_numerator
        End Get
        Set(numerator As Integer)
            m_numerator = numerator
        End Set
    End Property
    Public Property Denominator() As Integer
        Get
            Return m_denominator
        End Get
        Set(denominator As Integer) 'do not allow denominator to be 0
            If denominator <> 0 Then
```

```
                m_denominator = denominator
            End If
        End Set
    End Property
    Public Sub New()
        m_numerator = 0
        m_denominator = 1
    End Sub
    Public Sub New(numerator As Integer, denominator As Integer)
        m_numerator = numerator
        If denominator <> 0 Then
            m_denominator = denominator
        Else
            m_denominator = 1
        End If
    End Sub
    Function Improper() As Boolean
        Return m_numerator >= m_denominator
    End Function
    Function ToDecimal() As Double
        Return m_numerator / m_denominator
        'no test for divide by zero because denominator can't be 0
    End Function
    Overrides Function ToString() As String
     Return "" + m_numerator.ToString + "/" + m_denominator.ToString
    End Function
End Class
```

## Explanation

Notice that m_numerator and m_denominator are both private, with default values of 0 and 1. Making these members private allows us to restrict the values assigned to them. In particular, we want to make sure that the denominator is never zero.

Each private member usually has a Get and Set function.

**New**: When we create a New Fraction, the constructor function is called. Our Fraction class has two constructors: **`New` ()** and **`New(numerator As Integer, denominator As Integer).`**

**Improper**: The Improper function is Boolean (true or false).

**ToDecimal**: This function returns the decimal value of the fraction. Notice that we do not have to test for divide by zero because by making the denominator private, we have insured that it can never be zero.

**ToString**: Every class has a ToString function. We want to write our own definition of what ToString returns. We must override that predefined function with our own definition that will return the numerator, a slash, and the denominator.

## Testing The Fraction Class

We can now test the class in the main program. Try several different **instances** of the Fraction class in Form_Load. Experiment to test all of the functionality of the class.

```
Public Class Form1

Private Sub Form1_Load(sender As Object, e As EventArgs) _
   Handles MyBase.Load
        Dim half As New Fraction
        'half.Numerator = 1
        'half.Denominator = 2
        Me.Text = half.ToString
        'Me.Text = half.ToDecimal
        Dim third As New Fraction(1, 3)
        Me.Text = third.ToString
    End Sub
End Class
```

## To Do:

Add a text box for the numerator and denominator. Add labels to display the string; the decimal value and whether the fraction is improper.

You could expand this class further. Add a function to reduce the class. You could add a function ToStringProper that would return "1 1/3" instead of "4/3" *(This is not an override.)*

## To Think About:

Every industry has objects that they deal with everyday: transactions, packages, customers, boxes, printers, etc. How would you define any of these things as classes?