Préambule

Les tests unitaires (de classe et/ou de méthode...) permettent de vérifier que la classe et/ou la méthode sont conformes aux spécifications.

Dans la méthode de développement TDD [1], les tests unitaires sont écrits **avant** de coder les classes.

Les environnements de développement (*Visual Studio* par exemple) offrent des générateurs (partiels) de code de tests unitaires.

1 Tests unitaires de la classe `Fraction`

1.1 Introduction

La classe `Fraction` représente une fraction avec les différents constructeurs et méthodes permettant :

- de construire une fraction avec 0, 1, 2 paramètres
- de réduire automatiquement la fraction
- d'afficher la fraction (sous forme de chaîne, de nombre réel)
- d'effectuer des calculs entre fractions

La classe `Fraction` fournira les propriétés, constructeurs et méthodes du diagramme de la figure 1.

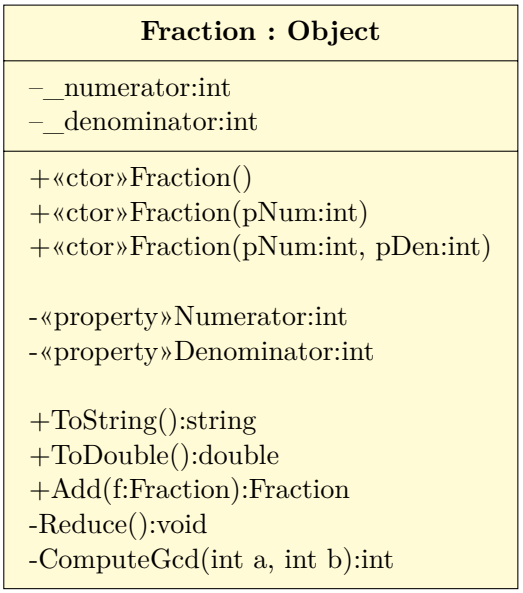

Figure 1 – Diagramme de la classe `Fraction`

1. *Test Driven Development*

1.2 Génération des tests unitaires

Remarque : il n'est possible de générer des tests unitaires **QUE** pour les classes `public`.

- clic-droit sur le nom de la classe à tester, « *Créer des tests unitaires* »
- conservez les valeurs par défaut
- OK
- un nouveau projet a été créé, il contient un fichier `FractionTests.cs`, voir listing 1

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Ex3Fraction.Tests {
  [TestClass()]
  public class FractionTests {
    [TestMethod()]
    public void FractionTest() {
      Assert.Fail();
    }

    [TestMethod()]
    public void FractionTest1() {
      Assert.Fail();
    }

 ...

    [TestMethod()]
    public void ToStringTest() {
      Assert.Fail();
    }
  }
}
```

Listing 1 – `FractionTests.cs`

1.3 Structure d'un test unitaire

Un test unitaire est composé de 3 parties :

1.3.1 Organiser (*Arrange*)

Création de l'objet à tester.

1.3.2 Agir (*Act*)

Appel d'une méthode ou modification d'un champ.

1.3.3 Vérifier (*Assert*)

Vérification du résultat.

1.3.4 Exemple

```
[TestMethod()]
public void ToStringTest() {
  // Arrange
  Fraction target = new Fraction(2, 3);

  // Act
  string answer = target.ToString();

  // Assert
  Assert.AreEqual("2 / 3", answer);
}
```

Listing 2 – Extrait de `FractionTests.cs`

1.4 Exécution des tests unitaires

- Menu *Test, Fenêtres, Explorateur de tests*
- dans la fenêtre « *Explorateur de tests* » :
  - cliquez sur le bouton « *Exécuter tout* »
  - la fenêtre « *Explorateur de tests* » visualise les résultats
  - la classe `Fraction` étant vide, certains tests devraient échouer ☹ mais d'autres (paradoxalement) devraient réussir ☺

2 Travail à effectuer

Écrivez les tests unitaires de la méthode `int ComputeGcd(int a, int b)` avec les données suivantes :

- $pgcd(0, 0) = 0$
- $pgcd(0, 1) = 1$
- $pgcd(0, 17) = 17$
- $pgcd(1, 1) = 1$
- $pgcd(10, 10) = 10$
- $pgcd(\texttt{Int32.MaxValue}, \texttt{Int32.MaxValue}) = \texttt{Int32.MaxValue}$
- $pgcd(\texttt{Int32.MaxValue}, 127) = 1$
- $pgcd(1000000007, 1000000009) = 1$
- $pgcd(-10, 10) = 10$
- $pgcd(-10, -10) = 10$
- $pgcd(23, 101) = 1$
- $pgcd(456, 456 * 123) = 456$

Indications : `https://fr.wikiversity.org/wiki/Arithmétique/PGCD#Extension_du_PGCD_aux_entiers_relatifs`

3 Compléments sur les tests unitaires

3.1 Accéder à un champ privé

`https://stackoverflow.com/questions/10789644/testing-a-private-field-using-mstest`

Ex : lecture du champ privé `_numerator` de la classe `Fraction` :

```
Fraction target = new FractionClass();
PrivateObject obj = new PrivateObject(target);

Assert.AreEqual(3, (int)obj.GetField("_numerator"));
Assert.AreEqual(8, (int)obj.GetField("_denominator"));
```

3.2 Tester une méthode privée

`https://stackoverflow.com/questions/9122708/unit-testing-private-methods-in-c-sharp`

Ex : test unitaire de méthode `private int ComputeGcd(int a, int b)` :

```
Fraction target = new FractionClass();
PrivateObject obj = new PrivateObject(target);

Assert.AreEqual(5,
                obj.Invoke("ComputeGcd", new object[] { 5, 15 })
               );
```

3.3 Détecter une exception

```
[TestMethod()]
[ExpectedException(typeof(StackEmpty), "StackEmpty.")]
public void PopEmptyStackTest()
{
  StackV3 target = new StackV3();
  target.Pop();
}
```

Listing 3 – Détecter une exception

```
class StackEmpty : Exception {
  public StackEmpty() : base("StackEmpty !") { }
}

class StackV3 {
  const int STACK_SIZE = 1000;
...
  public StackV3() { }
  public void Pop() { }
```

Listing 4 – Extraits de la classe `StackV3`

3.4 *Setup* et *TearDown*

Factorisation du code d'initialisation (*setup*, *initialize*) et de nettoyage (*teardown*, *cleanup*)

```
// Utilisez ClassInitialize pour exécuter du code avant ↩
   d'exécuter le premier test dans la classe
[ClassInitialize()]
public static void MyClassInitialize(TestContext testContext) {
}

// Utilisez ClassCleanup pour exécuter du code après que tous les ↩
   tests ont été exécutés dans une classe
[ClassCleanup()]
```

```
public static void MyClassCleanup() {
}

// Utilisez TestInitialize pour exécuter du code avant d'exécuter chaque test
[TestInitialize()]
public void MyTestInitialize() {
}

// Utilisez TestCleanup pour exécuter du code après que chaque test a été exécuté
[TestCleanup()]
public void MyTestCleanup() {
}
```

Listing 5 – *Setup* et *TearDown*

3.5 Tests unitaires paramétrables

Avec *Visual Studio 2017*, il est possible de paramétrer les tests unitaires afin d'utiliser une méthode de test avec différents jeux de paramètres.

```
using Microsoft.VisualStudio.TestTools.UnitTesting

[TestMethod()]
[DataRow(2, 3, "2 / 3")]
[DataRow(8, 16, "1 / 2")]
[DataRow(33, 11, "3")]
public void ToStringTest(int num, int den, string expectedFractionString) {
  // Arrange
  Fraction target = new Fraction(num, den);

  // Act
  string answer = target.ToString();

  // Assert
  Assert.AreEqual(expectedFractionString, answer);
}
```

Listing 6 – Extrait de `FractionTests.cs`

4 Références

- https://docs.microsoft.com/fr-fr/visualstudio/test/unit-test-basics
- http://c2.com/cgi/wiki?ArrangeActAssert
- https://msdn.microsoft.com/fr-fr/library/hh694602.aspx
- https://docs.microsoft.com/fr-fr/visualstudio/test/unit-test-basics