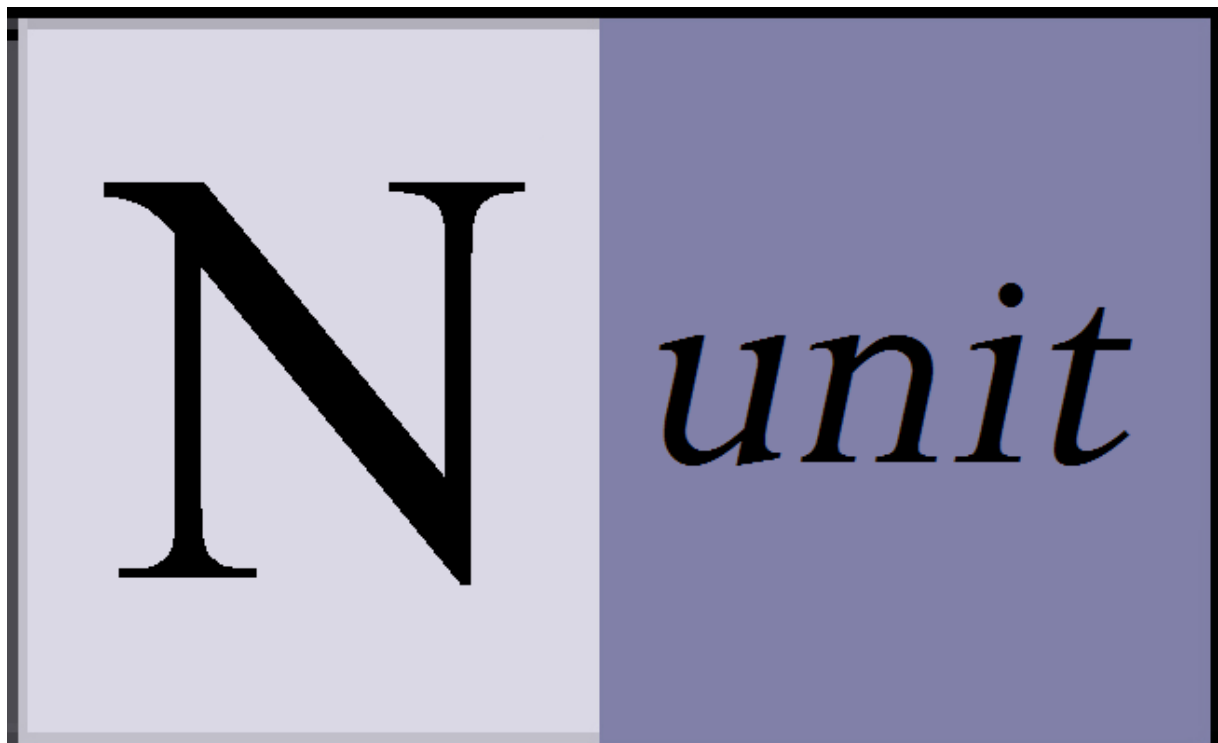


NUnit, un framework de test unitaire

Alexandre MIRABEL et Julien PERREIRA



Sommaire

I.	Présentation de NUnit.....	3
1.	Introduction.....	3
2.	Utiliser NUnit dans Visual Studio 2013.....	3
3.	Les annotations	5
II.	Exemples d'utilisation de NUnit	6
1.	Ouverture du projet et lancement des tests.....	6
2.	Utilisation des annotations [Test] et [TestFixture] sur le projet « Operator »	9
3.	Le projet « Operator » avec des tests fonctionnels.....	9
4.	Utilisation de l'annotation [ExpectedException] sur le projet « Operator »	9
III.	Glossaire	9
1.	Framework	9
2.	Annotations.....	10
IV.	Sources	10

I. Présentation de NUnit

1. Introduction

Un test unitaire est une méthode qui permet de tester un morceau de code d'un programme. Les données associées à un contrôle, les procédures d'utilisation et les procédures d'exploitation peuvent être testées. Ces tests ont pour but de savoir si une application est prête à l'emploi.

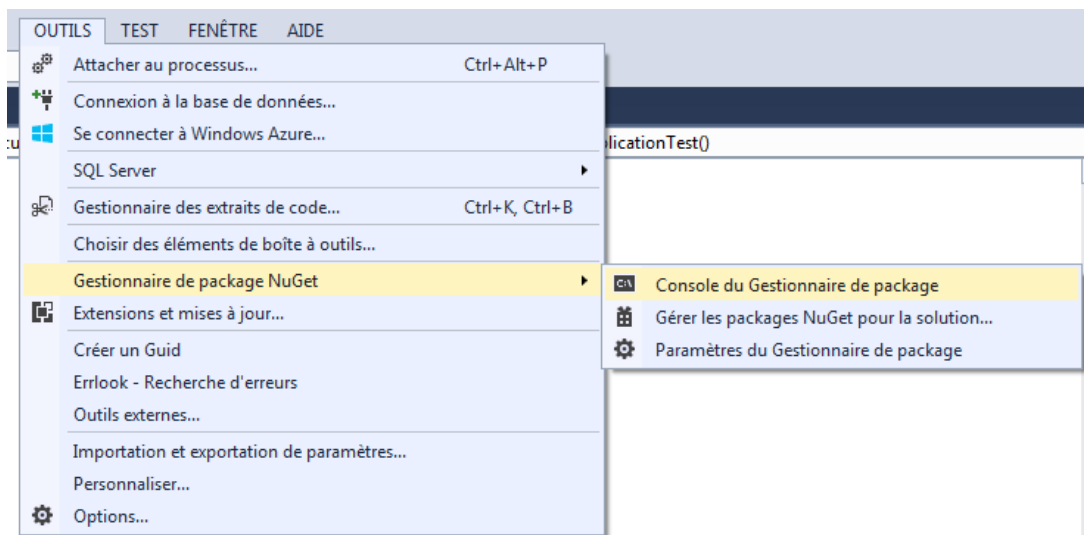
NUnit est un framework de test unitaire. Il peut-être téléchargé à l'adresse suivante : <http://nunit.org/index.php?p=download>

2. Utiliser NUnit dans Visual Studio 2013

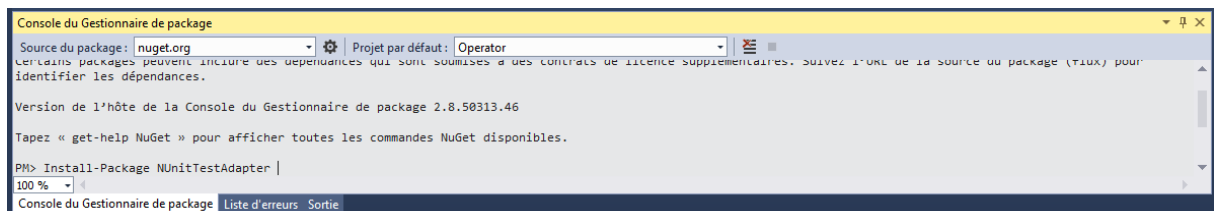
Pour utiliser NUnit, il faut utiliser une version de Visual Studio. Nous allons utiliser la version Express de 2013 de Visual Studio :

<http://www.microsoft.com/fr-fr/download/details.aspx?id=40787>

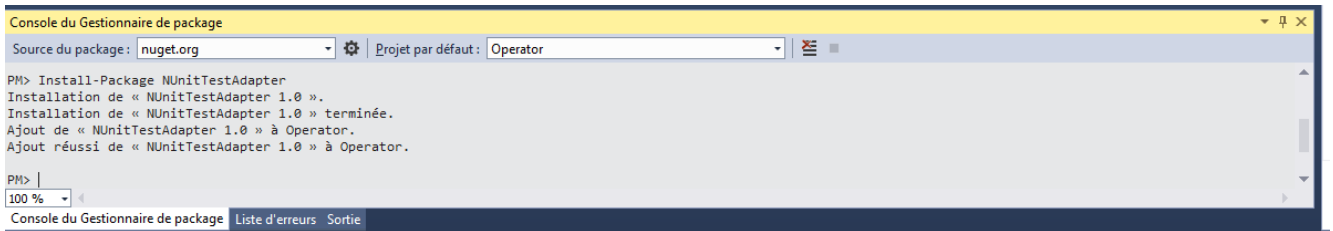
Ensuite, il faut installer un composant additionnel à Visual Studio 2013. Il faut cliquer sur le menu déroulant « Outils » et aller sur « Gestionnaire de package NuGet ». Ensuite, il faut cliquer sur « Console du Gestionnaire de package », comme ci-dessous :



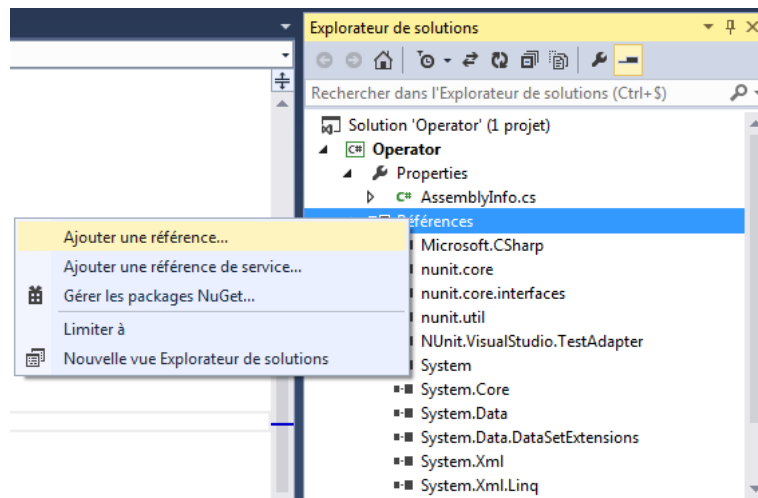
La console du gestionnaire de package s'ouvre et il faut écrire la commande « Install-Package NUnitTestAdapter » pour installer le composant NUnit pour Visual Studio 2013 :



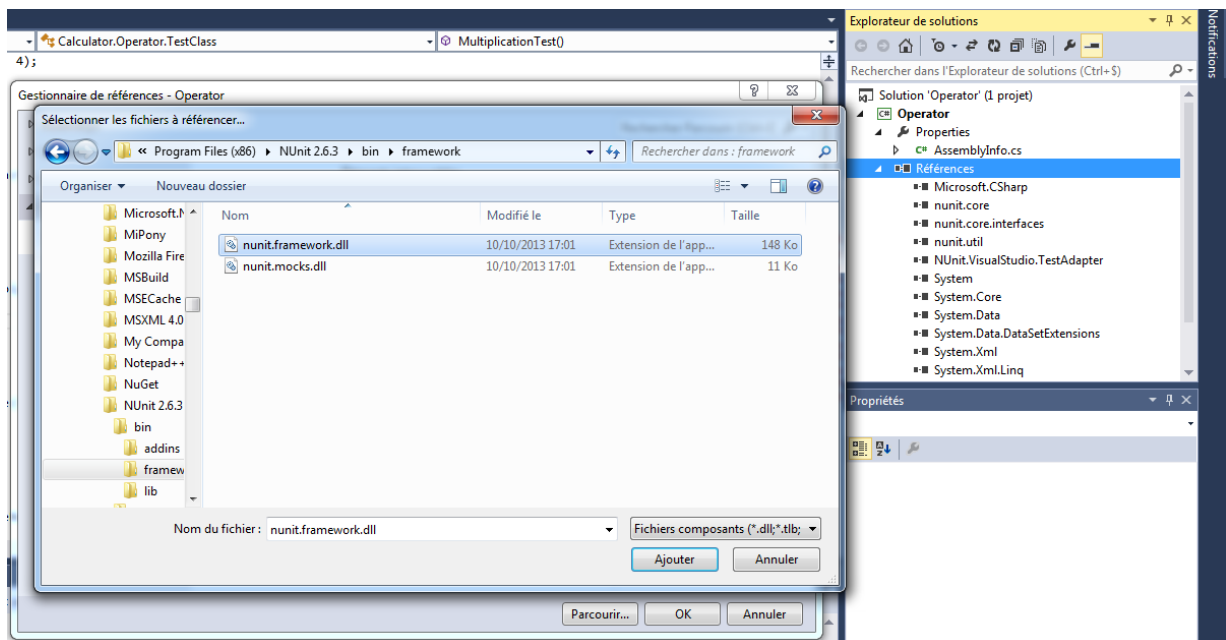
Le composant s'installe et il doit y avoir dans la console du gestionnaire de package les messages suivants :



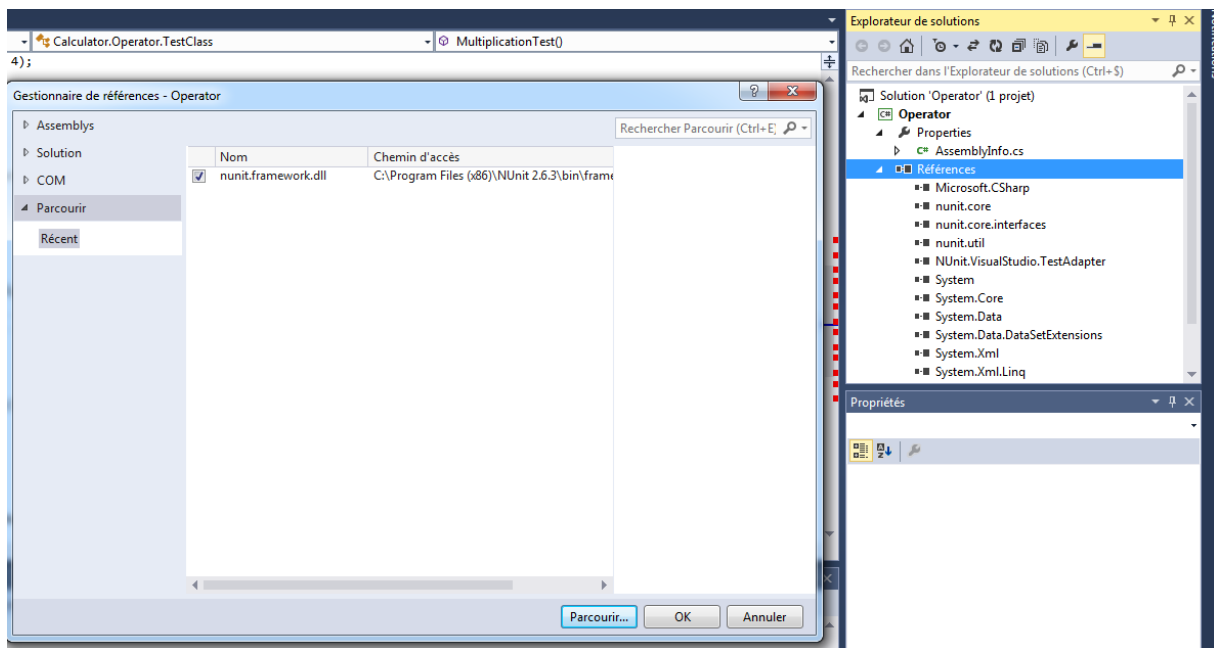
Enfin, dans la solution de Visual Studio 2013, il faut ajouter une référence au framework NUnit. Dans un premier temps, il faut faire un clic droit sur « Références » et cliquer sur « Ajouter une référence... »



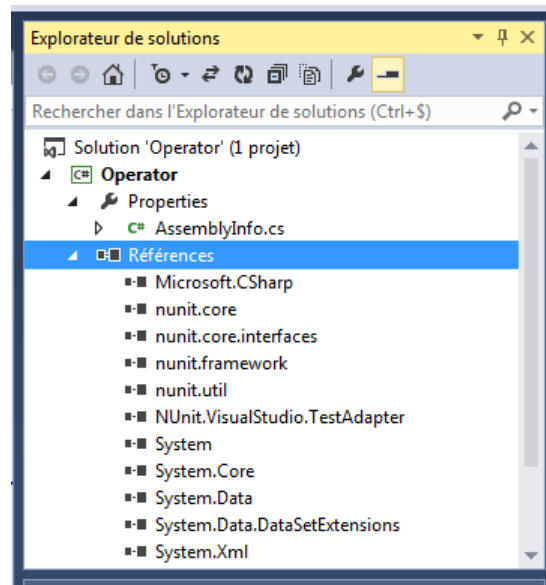
Il faut ajouter la dll « nunit.framework.dll » qui se trouve dans le dossier « C:\Program Files (x86)\NUnit 2.6.3\bin\framework » comme ci-dessous :



Ensuite, il faut cliquer sur le bouton « OK » et la référence est ajoutée à la solution comme ci-dessous :



Dans l'explorateur de solution, on peut vérifier que la dll a bien été ajoutée comme ci-dessous :



3. Les annotations

NUnit utilise les annotations pour savoir quelles méthodes sont des méthodes de test. Elles se notent au dessus de la définition d'une méthode. Les principales annotations utilisées sont les suivantes :

- [TestFixture] : Se place avant une classe et permet de savoir quelles classes sont des classes de test.
- [Before] : La méthode annotée sera exécutée avant la méthode annotée [Test].
- [Test] : Se place avant une méthode de test.

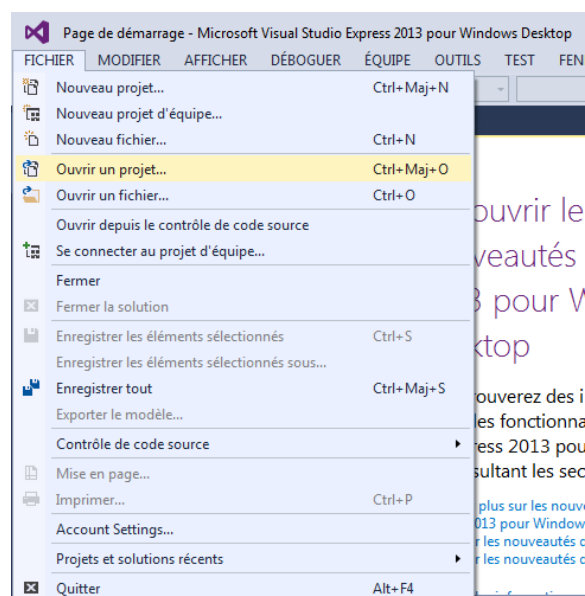
- [After] : La méthode annotée sera exécutée après la méthode annotée [Test].
- [ExpectedException(typeof(ArgumentException), ExpectedMessage="expected message")] : Le méthode doit renvoyé l'exception choisit. « typeof(ArgumentException) » est le type d'exception que l'on attend et « ExpectedMessage » est le message que l'on attend.

II. Exemples d'utilisation de NUnit

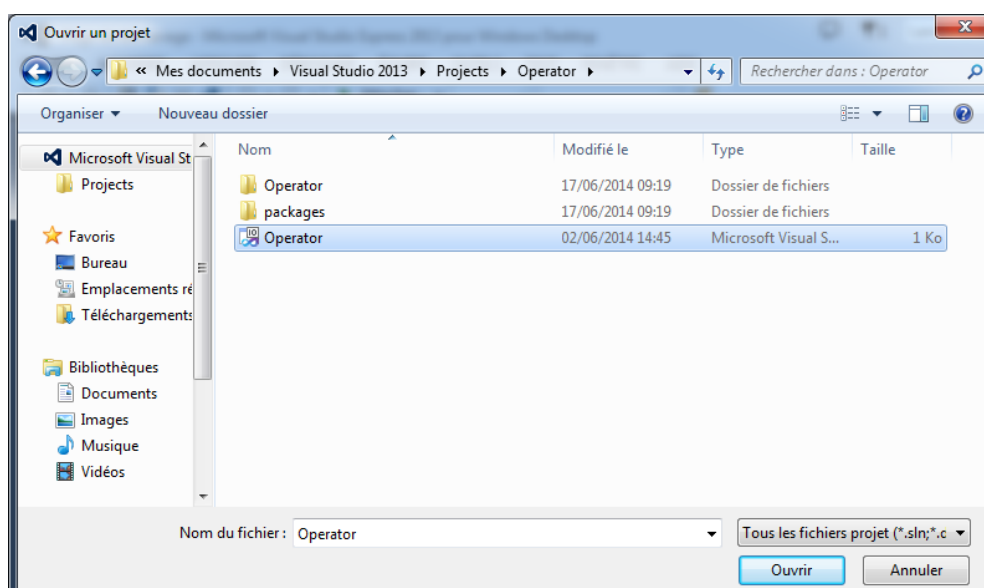
Dans cette partie, nous allons voir comment utiliser les annotations et créer des tests basiques à travers deux exemples (Sources et machine virtuelle fournit avec le tutoriel).

1. Ouverture du projet et lancement des tests

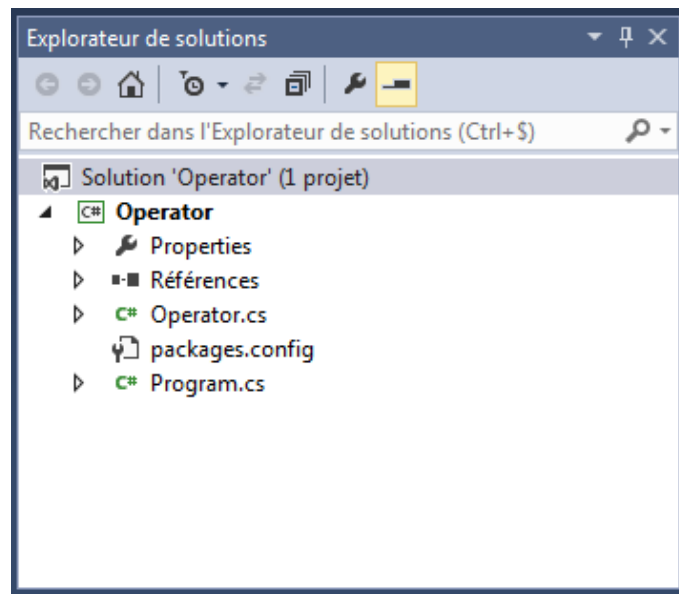
Pour les tests, nous avons utilisé Visual Studio 2013 Express. Pour ouvrir un projet, il faut aller dans « Fichier » et cliquer sur « Ouvrir un projet... » :



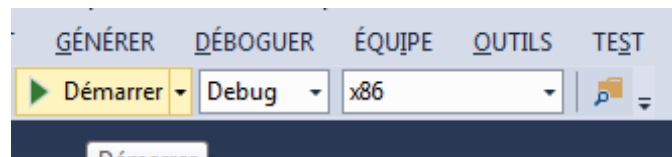
Ensuite, il faut aller dans le dossier du projet et cliquer sur le fichier « Operator » comme ci-dessous :



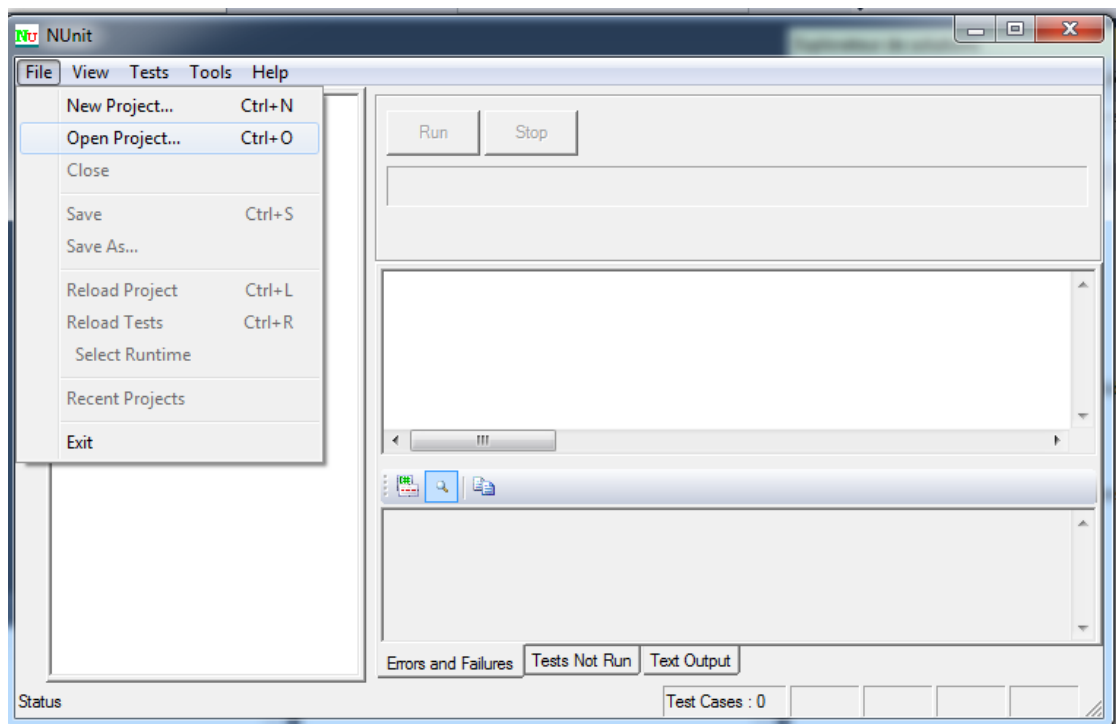
Lorsque le projet est ouvert, il doit apparaître dans l'explorateur de solution :



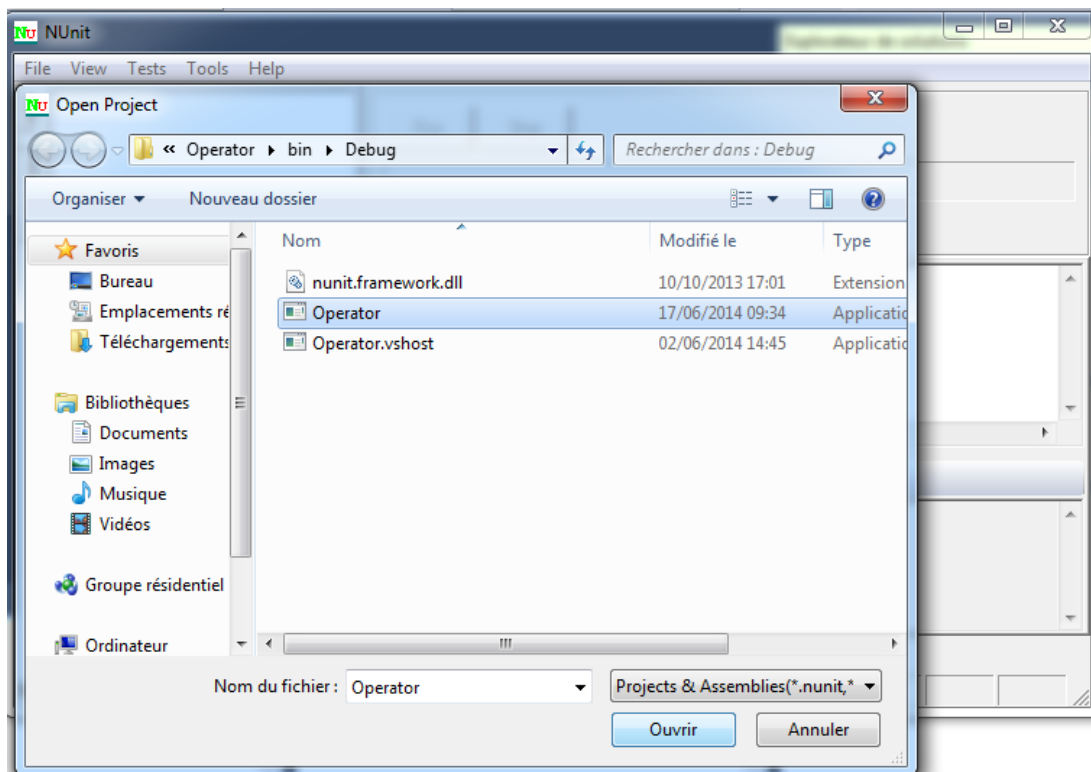
Ensuite, pour démarrer les tests, il faut cliquer sur le bouton « Démarrer » et être en mode « Debug » et en « x86 » :



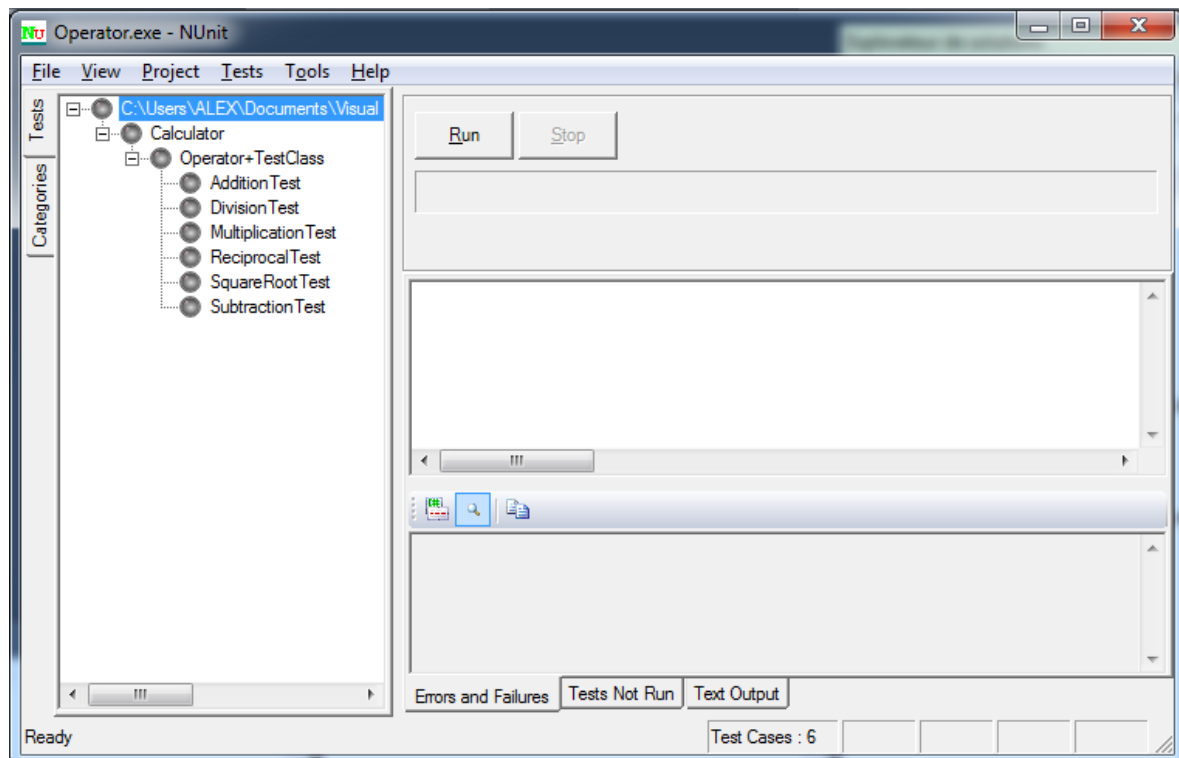
La fenêtre suivante s'affiche, et il faut cliquer sur « File » et sur « Open Project... » pour ouvrir le projet que nous voulons tester :



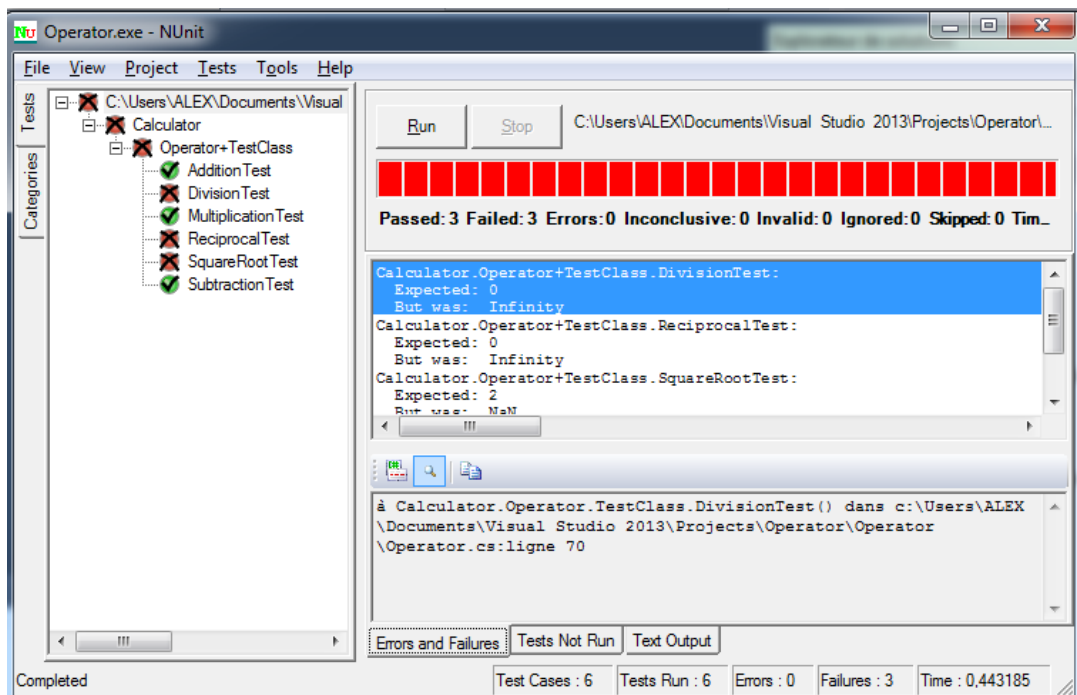
Il faut aller dans le dossier du projet puis dans le dossier « bin » et enfin dans le dossier « Debug » et choisir l'exécutable du projet (dans notre cas « Operator ») et cliquer sur « Ouvrir » :



Après ouverture du projet, on peut voir tous les tests qui vont être lancés. Pour lancer les tests, il faut cliquer sur le bouton « Run » :



A la fin des tests, la fenêtre change comme ci-dessous :



Ici, on voit les tests qui ont réussi et ceux qui ont échoué. De plus, on peut savoir pourquoi un test n'est pas passé.

2. Utilisation des annotations [Test] et [TestFixture] sur le projet « Operator »

Dans le projet « Operator » du dossier « Operator », nous avons une classe qui implémente les méthodes des principales opérations faites sur les nombres. Dans cette classe, nous avons une classe qui teste ces méthodes. Les tests ne fonctionnent pas tous car des cas n'ont pas été gérés.

3. Le projet « Operator » avec des tests fonctionnels

Le dossier « Operator2 » contient le même projet mais cette fois-ci les tests fonctionnent. Les tests fonctionnent car lorsque les valeurs en entrée sont incorrectes, le code retourne une valeur aberrante.

4. Utilisation de l'annotation [ExpectedException] sur le projet « Operator »

Le dossier « Operator3 » contient le même projet avec des tests fonctionnels et un code plus propre qui lève des exceptions au lieu de retourner des valeurs aberrantes.

III. Glossaire

1. Framework

Un framework est un ensemble cohérent de composants logiciels qui permette de créer les fondations et les grandes lignes de tout ou d'une partie d'un logiciel. Ils sont conçus pour modéliser l'architecture des logiciels applicatifs, des applications web...

2. Annotations

Une annotation est une façon d'ajouter des méta-données à un code source. Elles sont utilisées dans les tests de logiciel pour dire au compilateur quelles méthodes et classes sont utilisés pour le test.

IV. Sources

<https://www.nuget.org/packages/NUnitTestAdapter/>

<http://docs.nuget.org/docs/start-here/using-the-package-manager-console>

<http://www.nunit.org/index.php?p=exception&r=2.6.3>

<http://fr.wikipedia.org/wiki/Framework>