

Setelah melaksanakan praktikum ini diharapkan mahasiswa dapat :

- ✓ Menggunakan library NUnit untuk melakukan unit testing
-

Teori Singkat

Testing merupakan salah satu fase penting yang harus dilakukan ketika membuat sebuah aplikasi. Testing biasanya dilakukan secara berbarengan pada saat menulis kode. Ada banyak cara dalam melakukan testing dan yang paling favorit biasanya adalah cara konvensional. Dengan menggunakan cara ini programmer desktop akan menggunakan aplikasi console/winform untuk melihat hasil tes dari sebuah unit (fungsi/method) sedangkan programmer web langsung menggunakan halaman web.

Manfaat Testing:

- ✓ Menemukan bug secepat mungkin, sehingga mudah untuk diperbaiki.
- ✓ Meminimalkan bug di kemudian hari yang biasanya ditemukan oleh klien.
- ✓ Menuntut kita untuk menggunakan konsep *separation of concerns* dalam pembuatan program.
- ✓ Dokumentasi kode program.

Jenis-jenis Testing

- ✓ Unit Testing

Testing di level paling detail dari aplikasi. Artinya, testing di level method/function.

- ✓ Integration Testing

Integration Testing adalah tes (unit testing) yang dilakukan secara berkelompok. Atau dengan kata lain unit testing yang terintegrasi dengan sistem atau aplikasi lainnya.

- ✓ Coverage Testing

Tes yang dilakukan untuk mengecek apakah unit testing yang dibuat telah meng-cover semua kode yang perlu dites.

- ✓ Code Compliance Testing

Tes yang dilakukan untuk mengecek apakah penulisan kode sudah sesuai dengan aturan yang sudah ditetapkan.

- ✓ Performance Testing

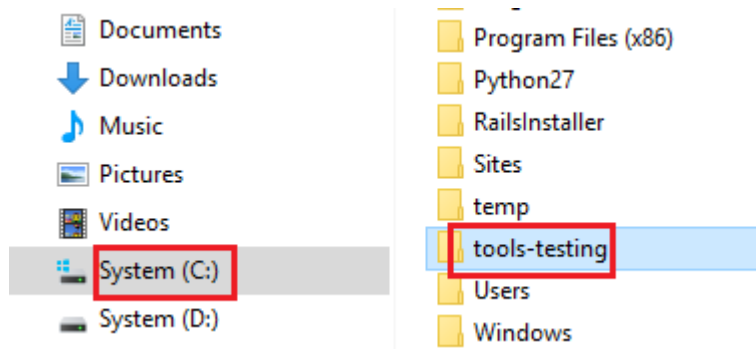
Performance testing adalah tes yang dilakukan untuk mengetahui seberapa cepat kinerja dari sebuah sistem di bawah beban kerja tertentu.

- ✓ User/Functional Testing atau UAT (User Acceptance Test)

Tes terakhir sebelum rilis ke production yang dilakukan langsung oleh user/client.

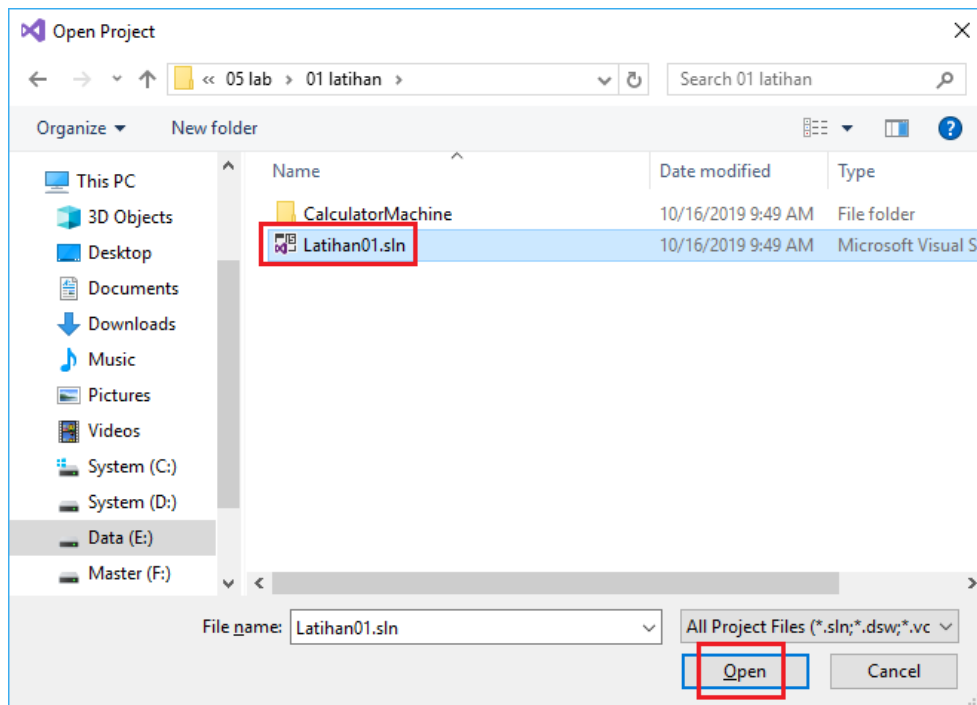
Persiapan Sebelum Praktikum

Copykan folder tools-testing ke drive C:\

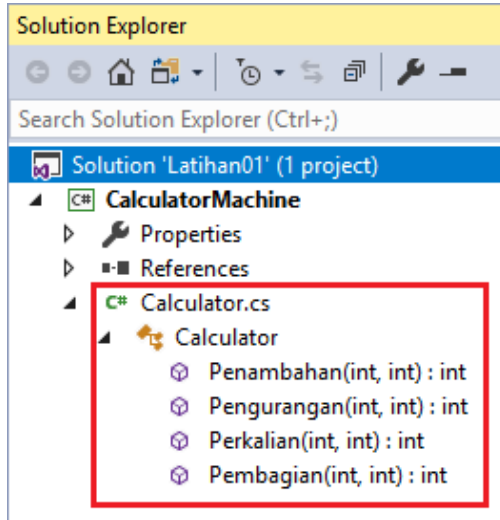


Latihan 5.1 (Unit Testing Menggunakan Aplikasi Console)

1. Awali selalu pekerjaan dgn doa, mudah-mudahan diberi kemudahan dan dapat memberikan manfaat
2. Jalankan aplikasi Microsoft Visual Studio .NET
3. Buka file solution Latihan01 yang sudah disertakan pada modul praktikum. Melalui menu File -> Open -> Project/Solution ...

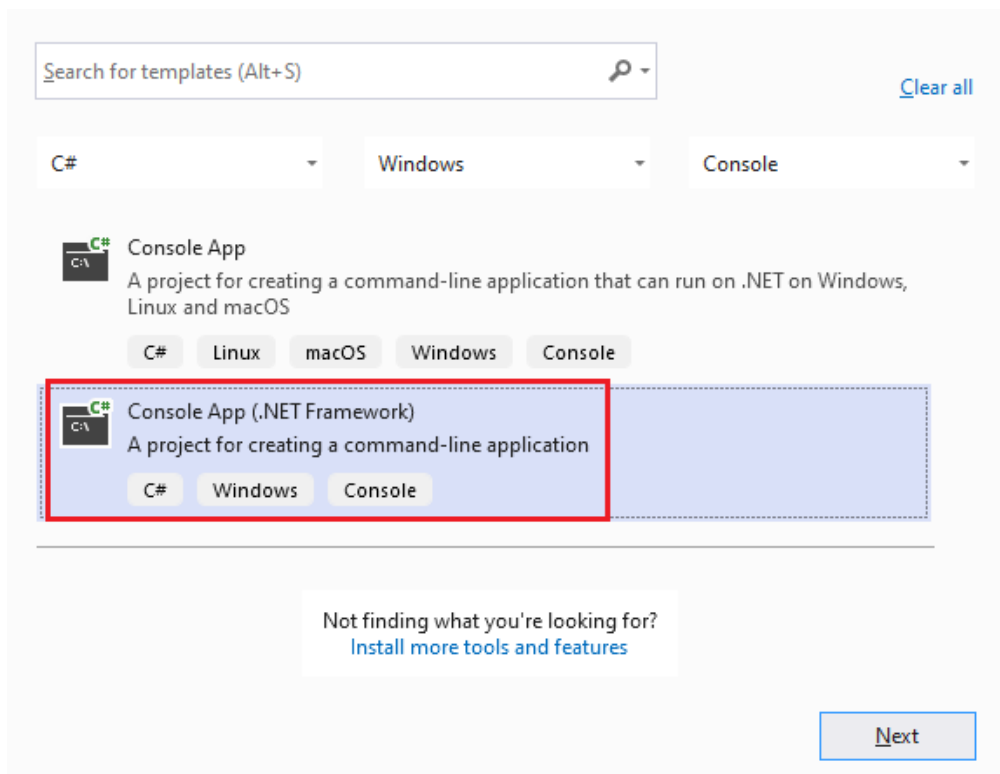


Pada solution ini sudah disertakan satu buah project Class Library dengan nama CalculatorMachine. Dan didalam project ini juga sudah terdapat sebuah class dengan nama Calculator yang mempunyai 4 buah method yaitu *Penambahan*, *Pengurangan*, *Perkalian* dan *Pembagian*.



Kita akan melakukan unit testing untuk semua method ini, baik unit testing yang dilakukan secara manual menggunakan aplikasi console atau menggunakan tool khusus untuk melakukan unit testing yaitu NUnit.

4. Tambahkan project baru (Console App) di solution yang sama dengan nama CalculatorConsoleTest



Configure your new project

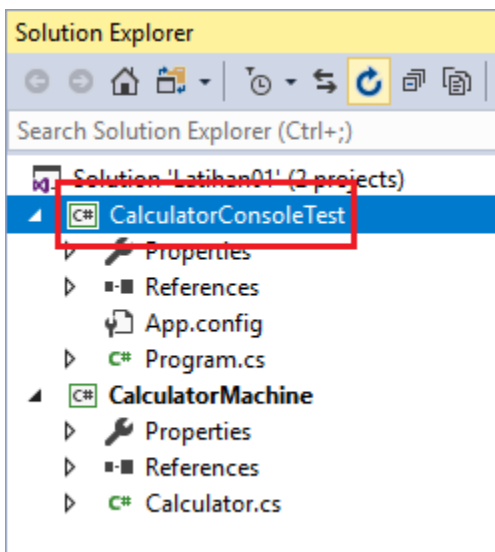
Console App (.NET Framework) C# Windows Console

Project name
CalculatorConsoleTest

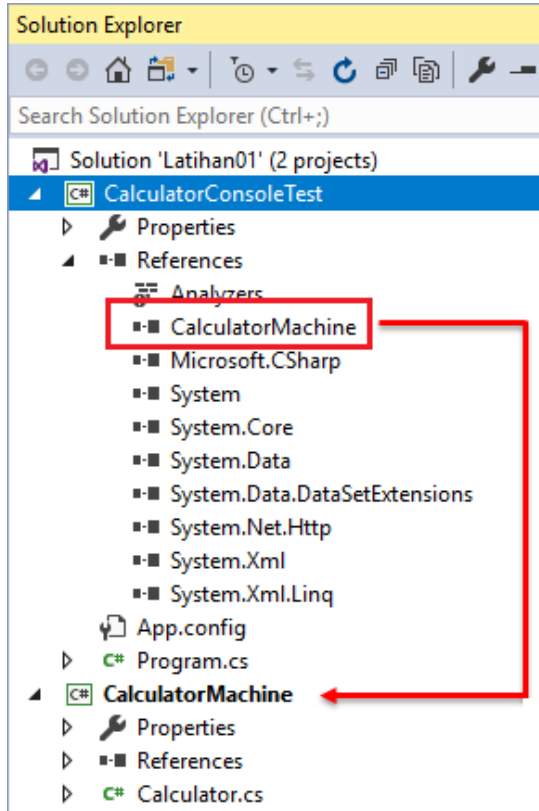
Location
E:\unit testing\01 project latihan

Framework
.NET Framework 4.6.1

Back Create

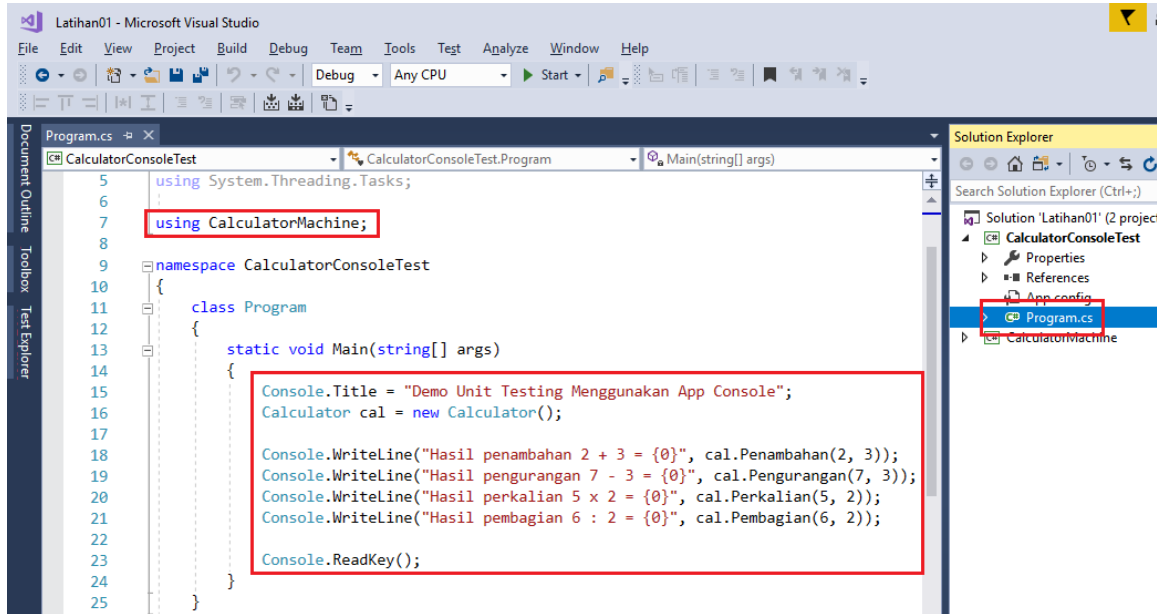


Setelah itu tambahkan referensi ke project CalculatorMachine, agar project CalculatorConsoleTest bisa mengakses semua class dan method public yang ada pada project CalculatorMachine.

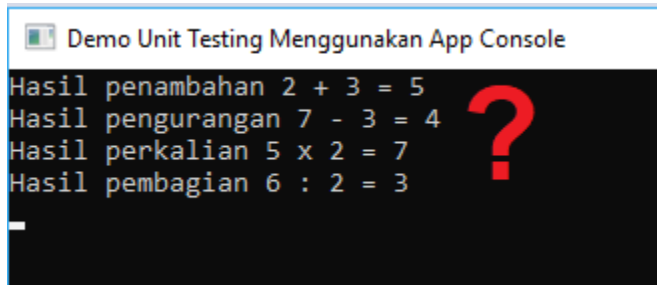


Jika Anda lupa langkah-langkah menambahkan referensi project ini, silahkan buka module praktikum 02.

Setelah itu lengkapi kode untuk method Mainnya seperti berikut:



5. Kemudian set project CalculatorConsoleTest sebagai startup project, dan tekan tombol F5 (Start Debugging) untuk menjalankan aplikasi.



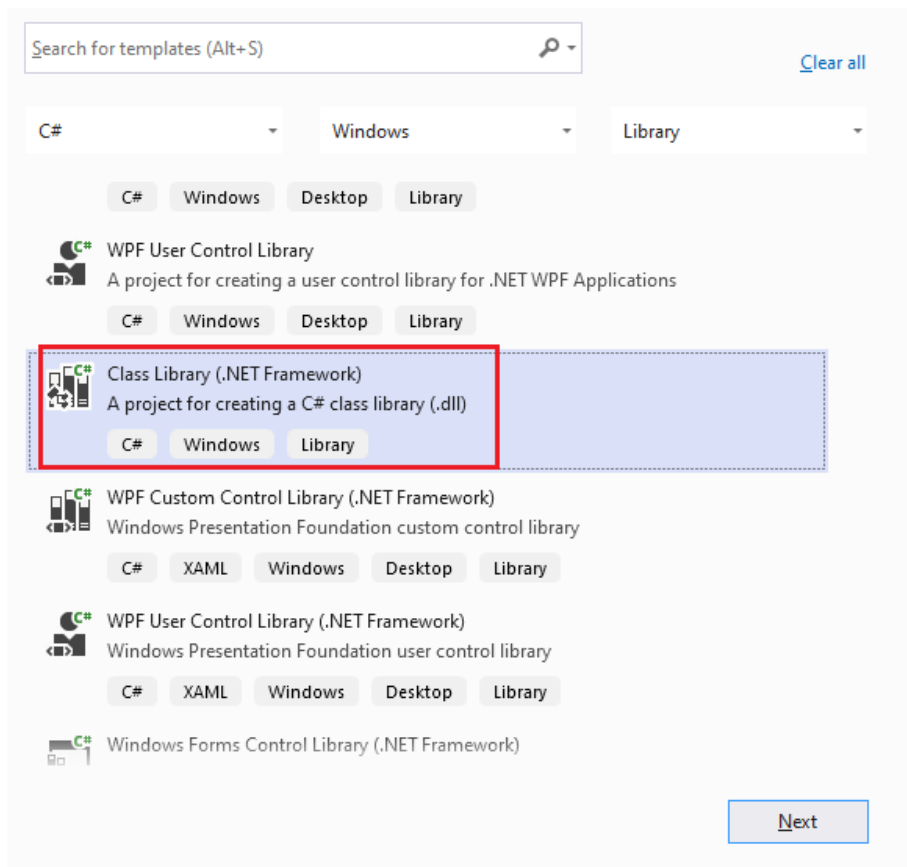
```
Demo Unit Testing Menggunakan App Console
Hasil penambahan 2 + 3 = 5
Hasil pengurangan 7 - 3 = 4
Hasil perkalian 5 x 2 = 7
Hasil pembagian 6 : 2 = 3
```

Cek output aplikasi, apakah sudah benar? Jika belum apa yang harus diperbaiki agar outputnya benar.

Latihan 5.2 (Unit Testing Menggunakan Library NUnit)

Setelah melakukan unit testing dengan cara jadul (konvensional), saatnya kita menggunakan library yang memang dirancang khusus untuk melakukan unit testing yaitu NUnit. Dengan library ini kita sudah tidak perlu lagi mengevaluasi hasil tes secara manual karena komputerlah yang akan melakukannya.

1. Masih di solution yang sama, tambahkan project baru (Class Library) dengan nama CalculatorUnitTest



Configure your new project

Class Library (.NET Framework)

C#

Windows

Library

Project name

CalculatorUnitTest

Location

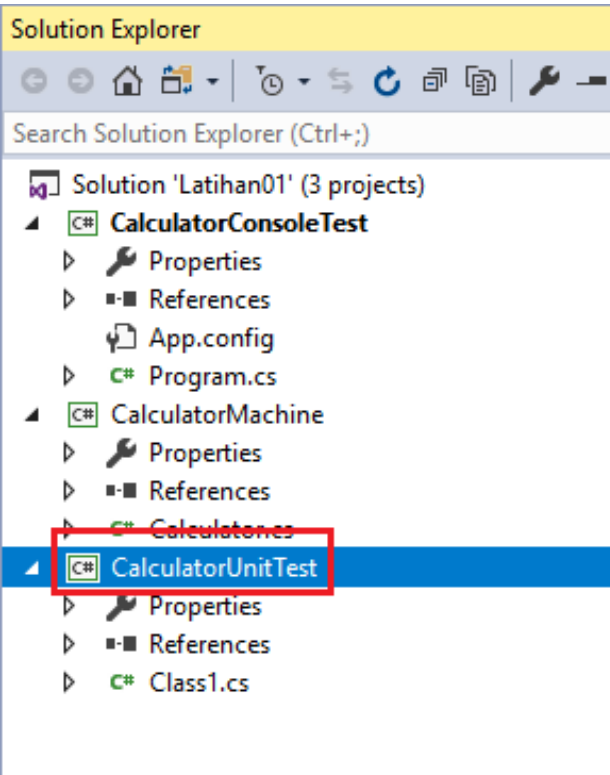
E:\Unit Testing\01 project latihan

Framework

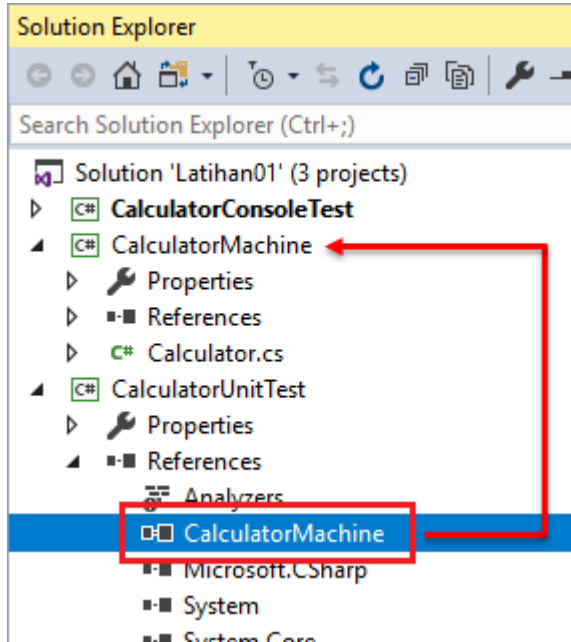
.NET Framework 4.6.1

Back

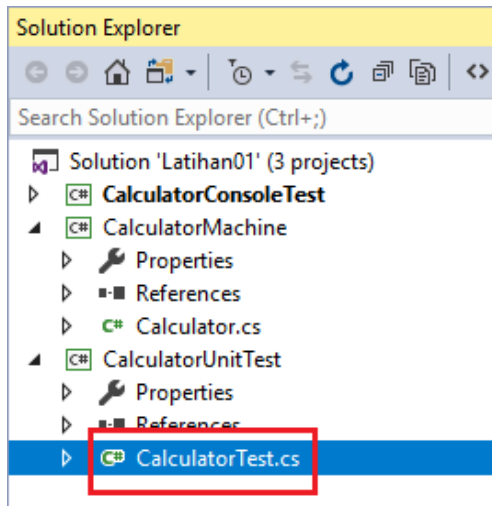
Create



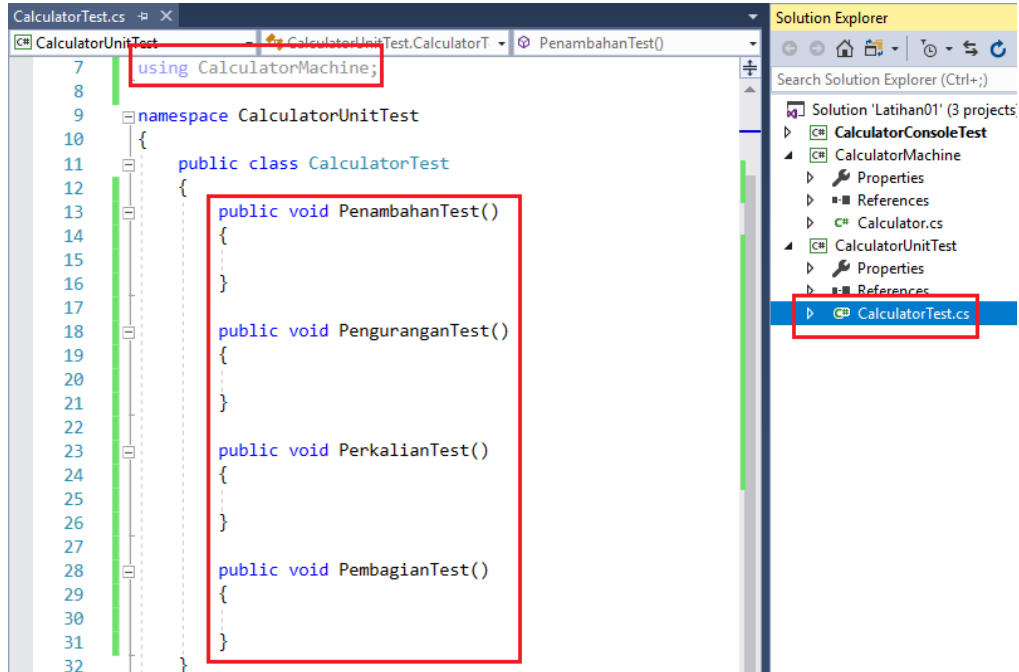
Kemudian tambahkan juga referensi ke project CalculatorMachine



Setelah itu tambahkan class baru dengan nama CalculatorTest. Di dalam class ini kita akan menuliskan kode untuk unit testingnya.

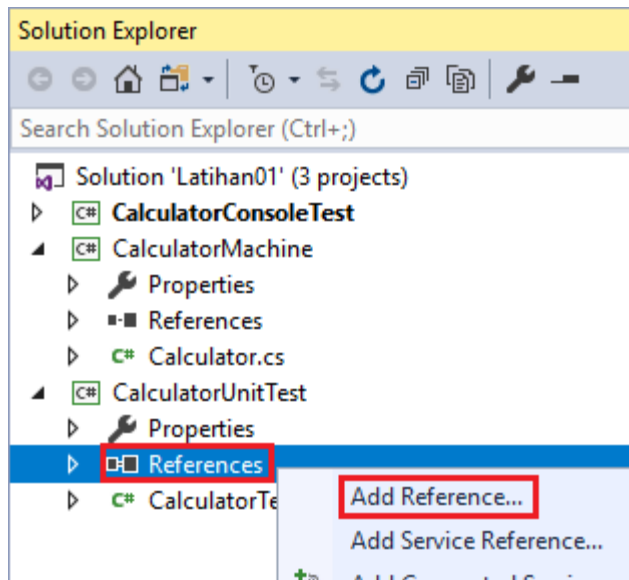


Kemudian tambahkan 4 buat method tes seperti berikut:

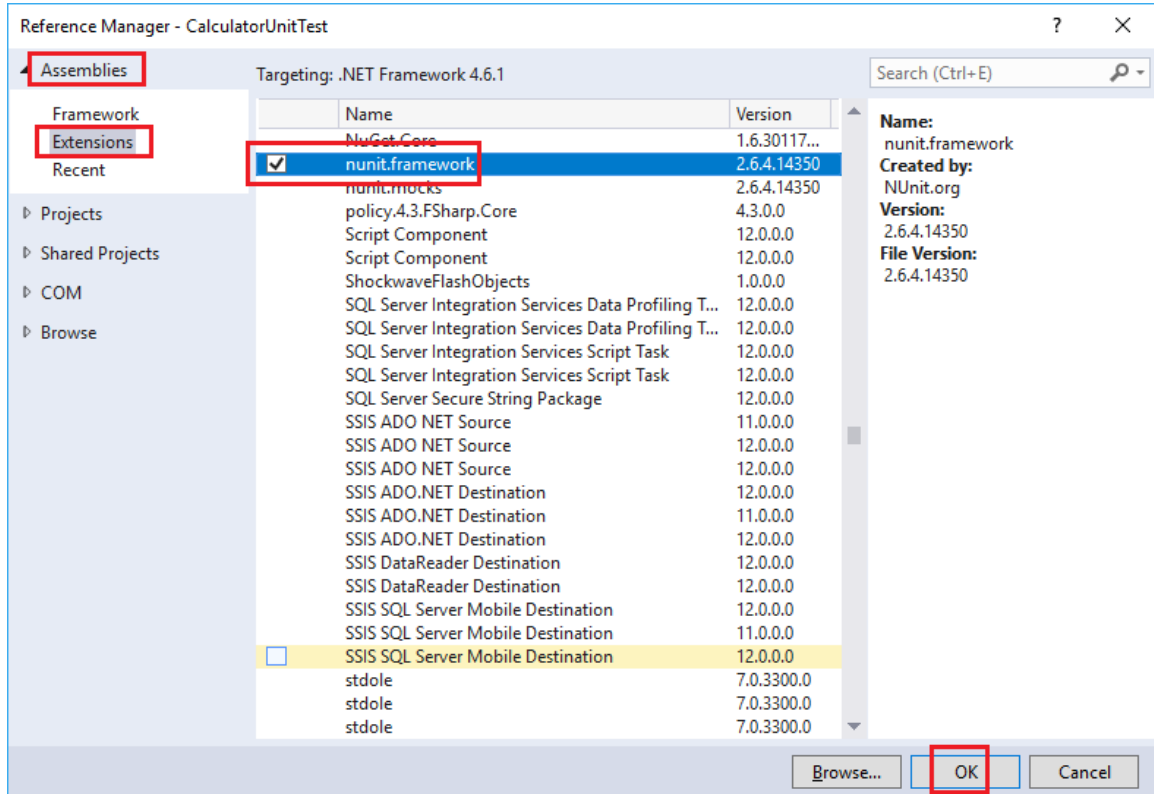


Method-method yang ada di class unit testing, biasanya bersifat independent artinya masing-masing method ini bisa dijalankan/eksekusi secara mandiri (tidak tergantung dengan method yang lain) sehingga kita bisa menjalankan unit testing secara berulang.

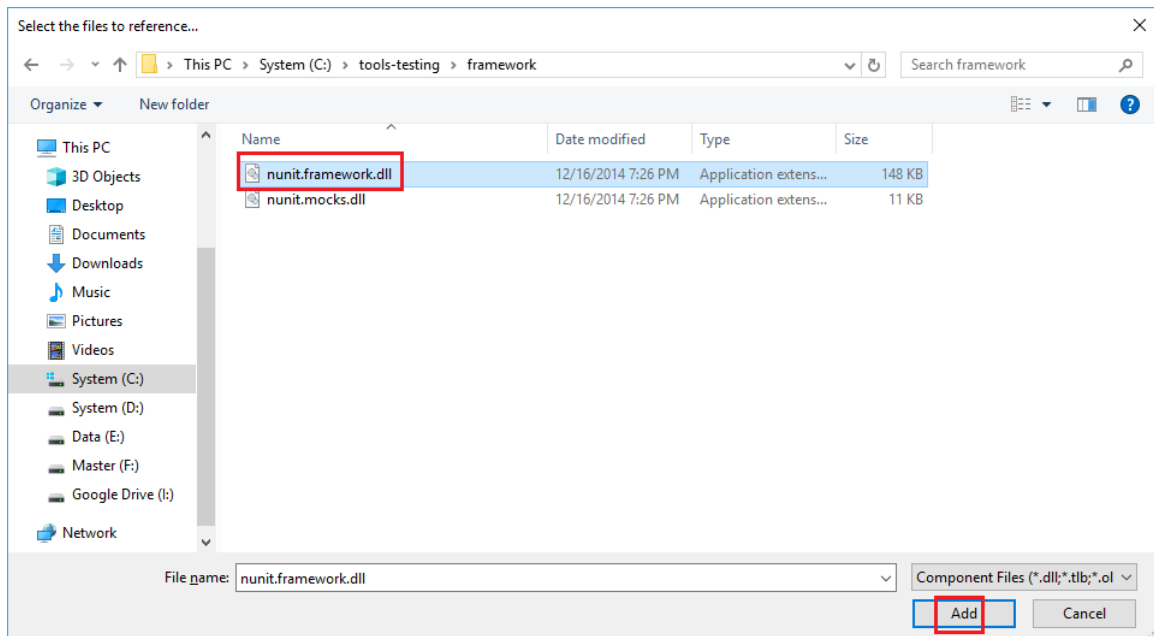
- Langkah berikutnya adalah menambahkan referensi library NUnit dengan cara klik kanan node References -> Add Reference..



- Setelah tampil dialog Reference Manager, aktifkan panel Assemblies -> Extensions kemudian aktifkan pilihan nunit.framework

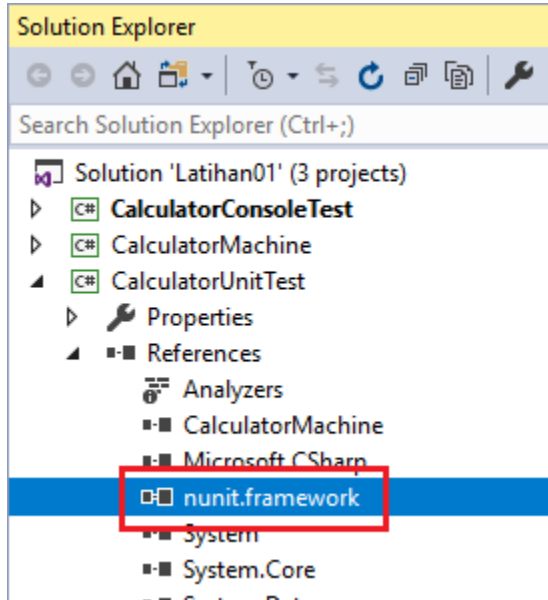


Jika pilihan library *nunit.framework* belum tersedia, silahkan Anda klik tombol *Browse...*, kemudian arahkan ke folder *C:\tools-testing\framework* dan pilih file *nunit.framework.dll*



Klik tombol Add setelah itu akhiri dengan menekan tombol Ok.

Setelah melakukan langkah-langkah di atas, silahkan cek lagi node References, seharusnya sudah ada tambahan library *nunit.framework*

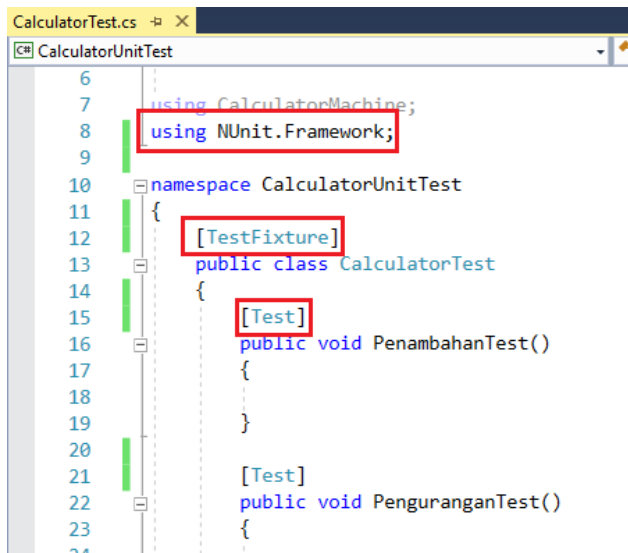


Setelah menambahkan library NUnit, project CalculatorUnit'Test tidak otomatis langsung dikenali oleh tool NUnit, ada hal-hal lainnya yang perlu kita ketahui dalam menulis unit testing menggunakan NUnit yaitu :

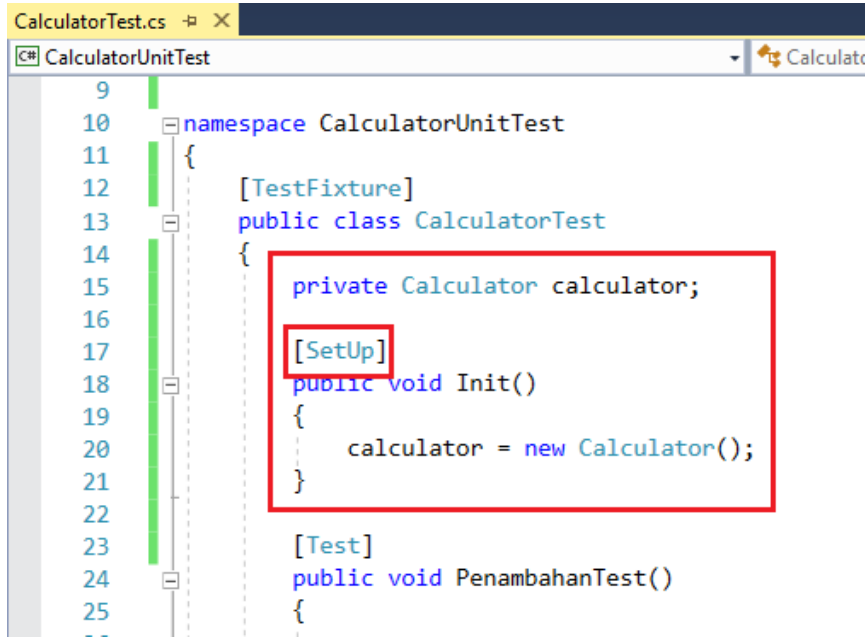
- Attributes
- Assertions

Attributes

Digunakan untuk memberi tag/penanda class dan method yang akan dites. Ada dua attribut yang sering digunakan yaitu attribut [TestFixture] untuk menandai class dan attribut [Test] untuk menandai method yang akan di tes. Contoh :



Selain itu ada attribut lain yang digunakan untuk menandai method yang berfungsi sebagai method inisialisasi yaitu attribut [SetUp]. Contoh :

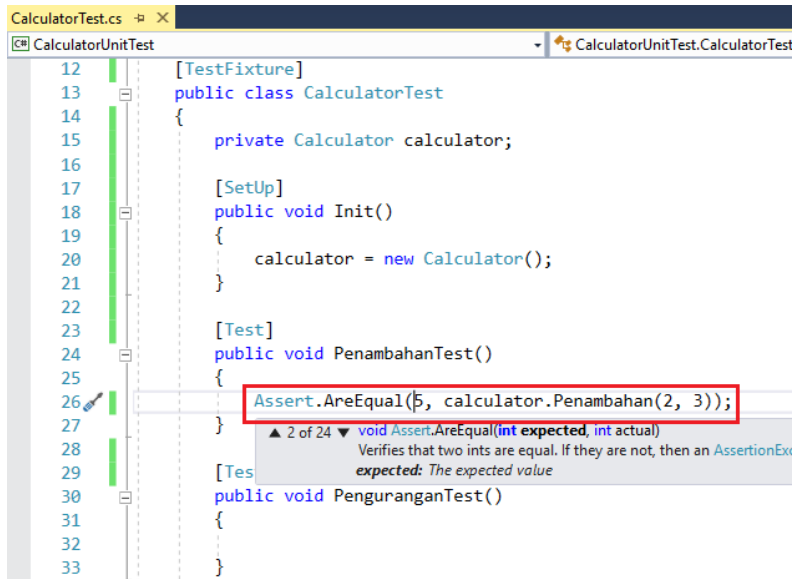


```
9
10 namespace CalculatorUnitTest
11 {
12     [TestFixture]
13     public class CalculatorTest
14     {
15         private Calculator calculator;
16
17         [SetUp]
18         public void Init()
19         {
20             calculator = new Calculator();
21         }
22
23         [Test]
24         public void PenambahanTest()
25         {
```

Jadi method *Init* di atas akan dijalankan terlebih dulu sebelum tool Nunit menjalankan method-method lain yang ditandai dengan attribut *[Test]*.

Assertions

Merupakan sekumpulan method static yang digunakan untuk mengevaluasi method yang akan dites. Contoh :



```
12 [TestFixture]
13 public class CalculatorTest
14 {
15     private Calculator calculator;
16
17     [SetUp]
18     public void Init()
19     {
20         calculator = new Calculator();
21     }
22
23     [Test]
24     public void PenambahanTest()
25     {
26         Assert.AreEqual(5, calculator.Penambahan(2, 3));
27     }
28
29     [Test]
30     public void PenguranganTest()
31     {
32     }
33 }
```

Pada gambar di atas kita menggunakan salah satu method Assertions yaitu method *AreEqual* untuk membandingkan dua buah nilai, yaitu nilai yang di harapkan (*expected*) dan nilai yang dihasilkan dari pemanggilan method (*actual*). Nilai *expected* kita inputkan secara manual sedangkan nilai *actual* dihasilkan dari pemanggilan method. Coba perhatikan potongan kode berikut :

```
[Test]
public void PenambahanTest()
{
    Assert.AreEqual(5, calculator.Penambahan(2, 3));
}

[Test]
public void PenguranganTest()
{
```

▲ 2 of 24 ▼ void Assert.AreEqual(int expected, int actual)
Verifies that two ints are equal. If they are not, then an Assertio
actual: The actual value

Dari hasil pemanggilan method `Penambahan(2, 3)`, jika menghasilkan nilai 5 maka method `AreEqual` akan menginformasikan bahwa tesnya berhasil karena sesuai dengan nilai yang diharapkan yaitu 5 selain itu akan gagal.

4. Berikut kode lengkap class `CalculatorTest` :

```
using CalculatorMachine;
using NUnit.Framework;

namespace CalculatorUnitTest
{
    [TestFixture]
    public class CalculatorTest
    {
        private Calculator calculator;

        [SetUp]
        public void Init()
        {
            calculator = new Calculator();
        }

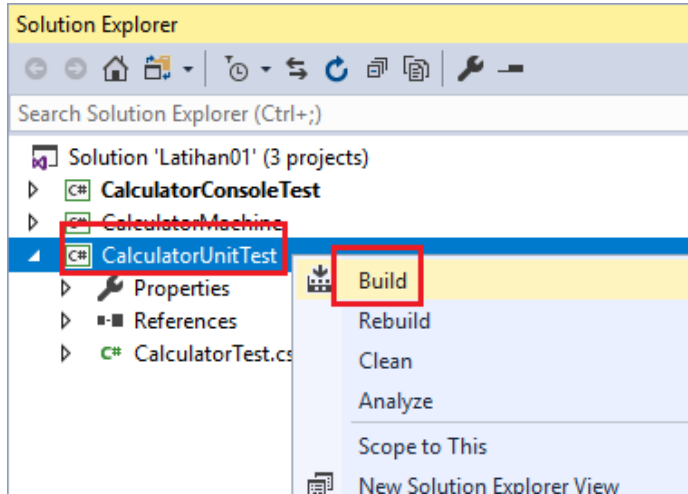
        [Test]
        public void PenambahanTest()
        {
            Assert.AreEqual(5, calculator.Penambahan(2, 3));
        }

        [Test]
        public void PenguranganTest()
        {
            Assert.AreEqual(4, calculator.Pengurangan(7, 3));
        }

        [Test]
        public void PerkalianTest()
        {
            Assert.AreEqual(10, calculator.Perkalian(5, 2));
        }

        [Test]
        public void PembagianTest()
        {
            Assert.AreEqual(3, calculator.Pembagian(6, 2));
        }
    }
}
```

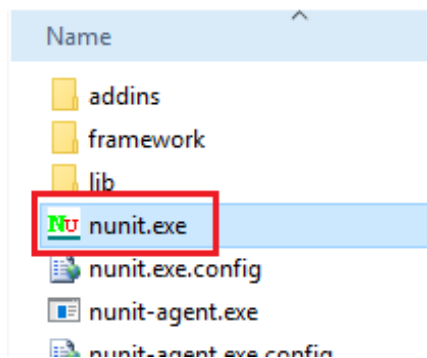
5. Setelah itu lakukan proses Build untuk project `CalculatorUnitTest`

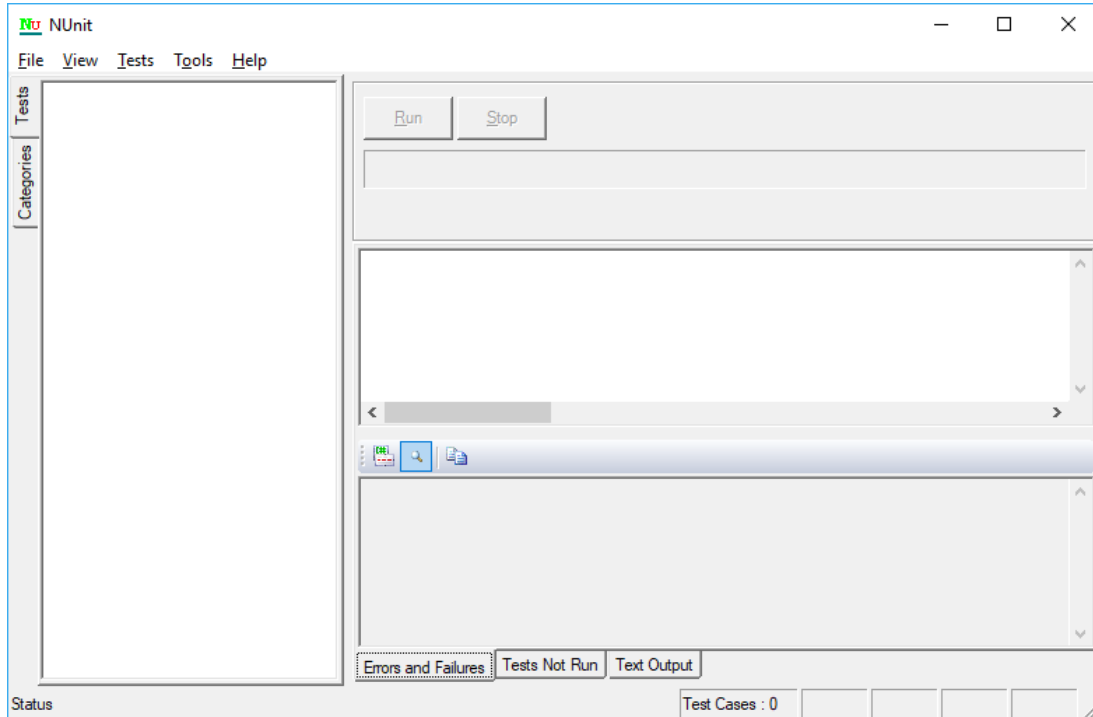


Dari proses build ini akan menghasilkan file assembly dengan ekstensi dll. Lokasinya ada di folder project CalculatorUnitTest\bin\Debug

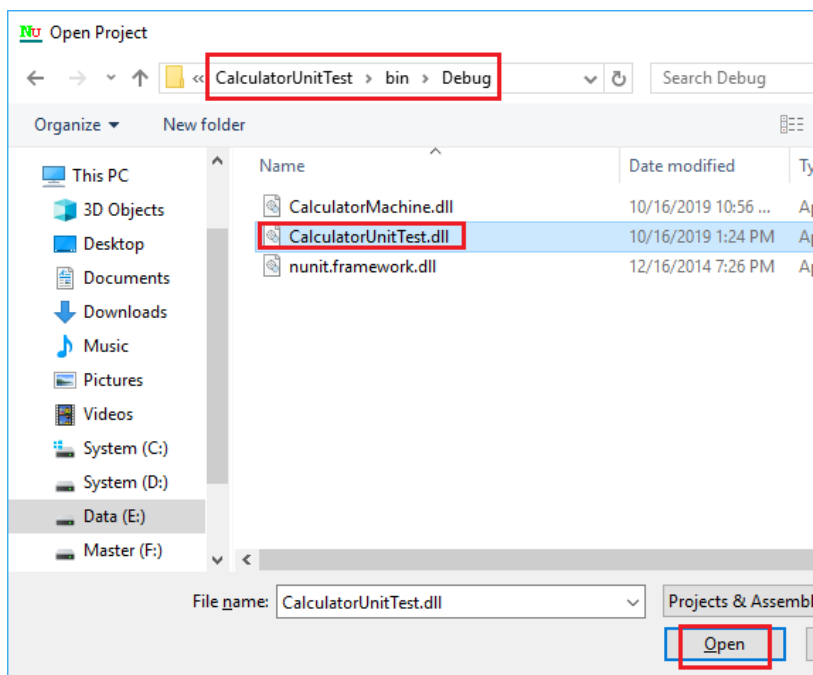
Name	Date mo
CalculatorMachine.dll	10/16/2023
CalculatorMachine.pdb	10/16/2023
CalculatorUnitTest.dll	10/16/2023
CalculatorUnitTest.dll.VisualState.xml	10/16/2023
CalculatorUnitTest.pdb	10/16/2023
nunit.framework.dll	12/16/2022

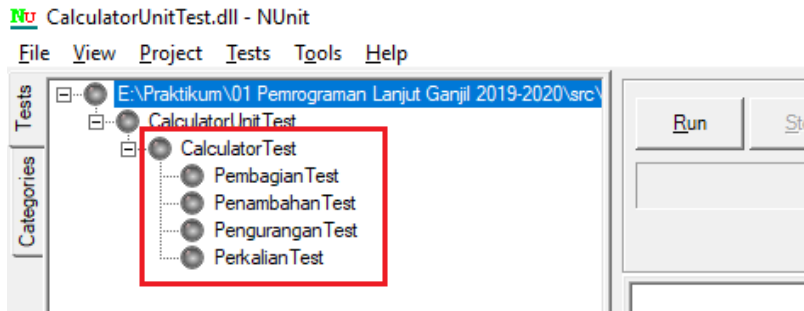
- Langkah berikutnya adalah menjalankan tool NUnit untuk menjalankan project unit testing. Tool ini membutuhkan file assembly (dll) yang dihasilkan dari proses build dari project unit testing (langkah no 5).
- Untuk menjalankan tool NUnit ini, jalan kan file nunit.exe yang ada di folder C:\tools-testing\NUnit



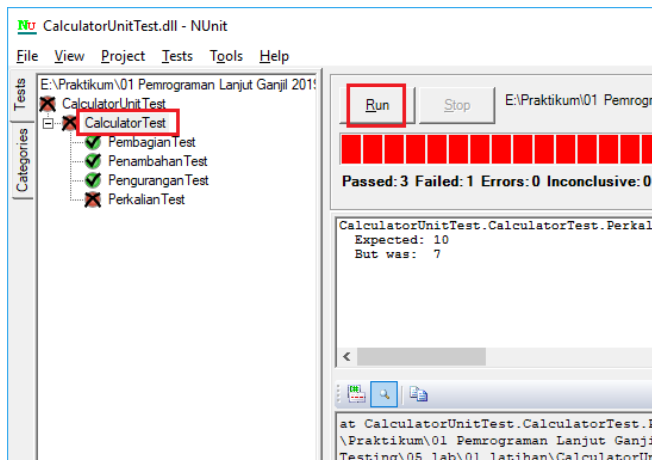


8. Setelah tool NUnit ini jalan, klik menu File -> Open Project, kemudian cari lokasi hasil build CalculatorUnitTest, yang ada di folder CalculatorUnitTest\bin\Debug





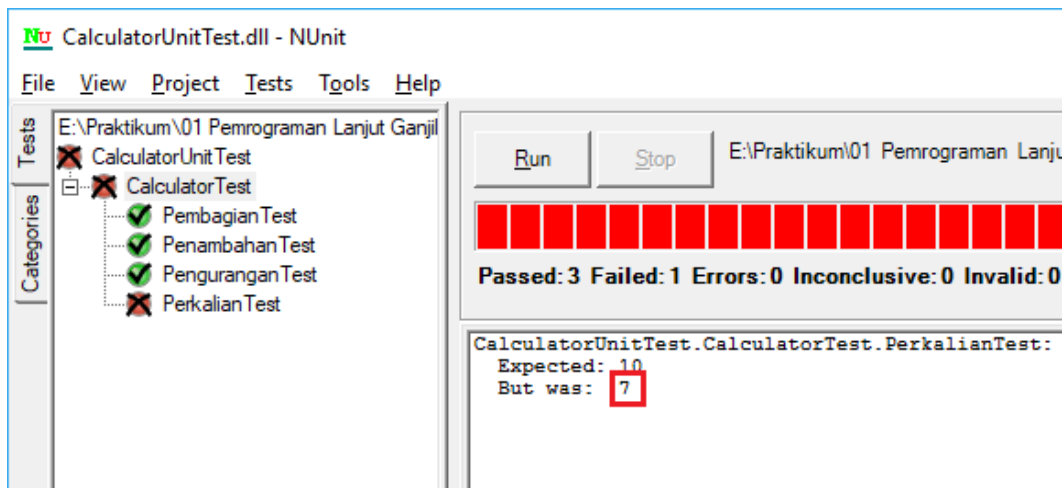
9. Jika ingin langsung mengecek hasil unit testingnya, klik tombol Run



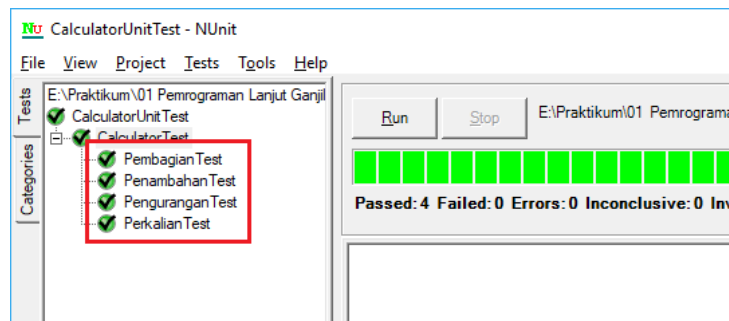
Pada gambar di atas hanya method Perkalian yang gagal melewati unit test. Coba kita cek kode untuk unit testingnya.

```
[Test]
public void PerkalianTest()
{
    Assert.AreEqual(10, calculator.Perkalian(5, 2));
}
```

Pada kode di atas 5×2 seharusnya menghasilkan nilai 10 (nilai expected/diharapkan), tetapi dari hasil unit testingnya malah menghasilkan nilai 7 (nilai actual). Nah yang jadi pertanyaan dari mana datangnya nilai 7 ini ?

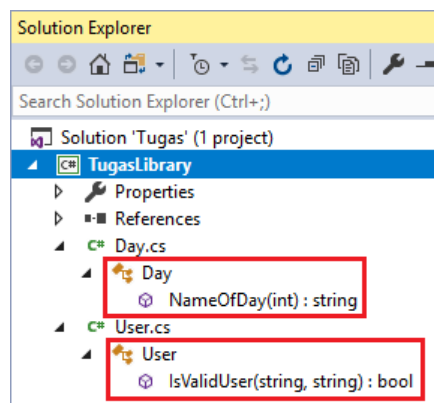


Dari sini kita sudah bisa menebak ada yang salah dengan method Perkaliannya, jadi silahkan cek dan perbaiki lagi method Perkaliannya (setelah itu jangan lupa di build ulang projectnya). Kemudian di tes lagi sampai hasil unit testingnya seperti berikut.

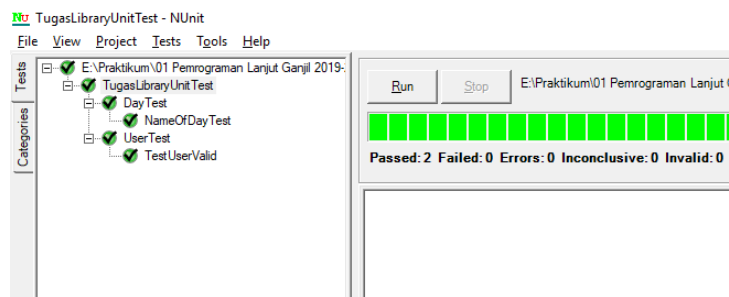


Tugas 5.1

- ✓ Buatkan project unit testing (Class Library) dengan nama TugasLibraryUnitTest untuk project TugasLibrary berikut:



- ✓ Kemudian tes dengan menggunakan tool NUnit.



Selesai 😊

Kamarudin, M.Kom
<http://coding4ever.net/>
<https://github.com/rudi-krsoftware/open-retail>