

Materi:

- ✓ Lanjut bekerja dengan multi project
- ✓ Shared Code Menggunakan Project Library

Pada praktikum kedua ini, kita masih melanjutkan praktikum yang pertama tentang pembuatan aplikasi calculator yang berbasis console dan desktop (windows form).

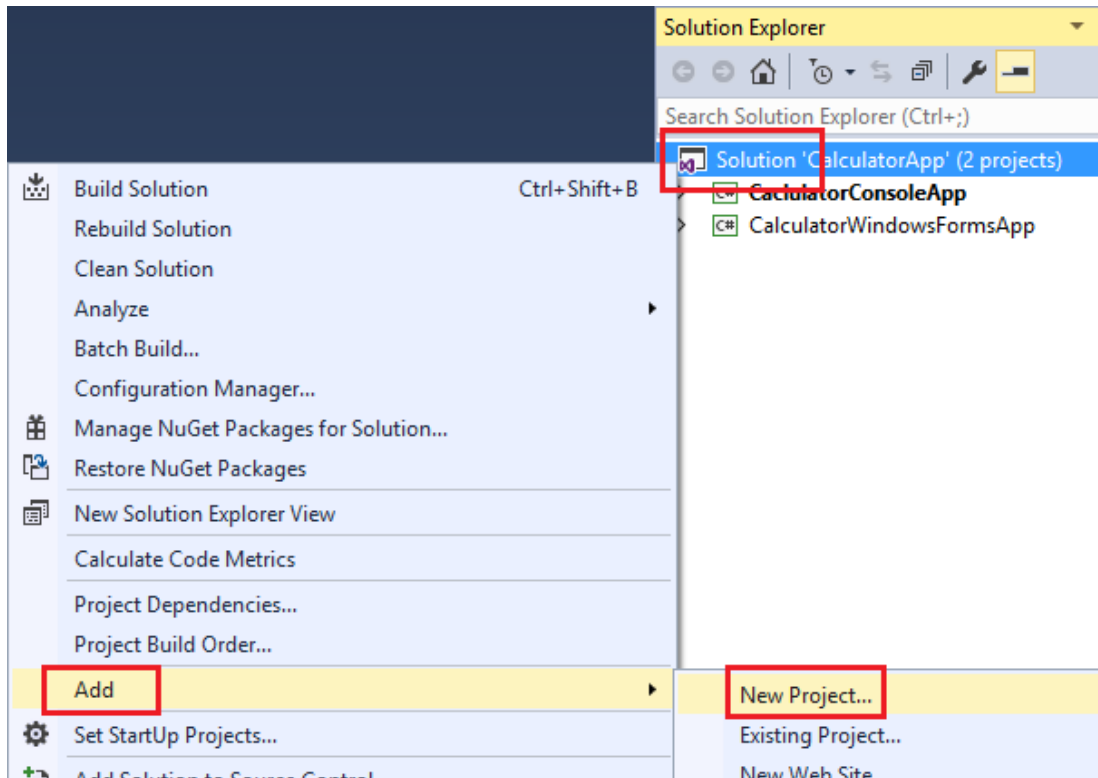
Kalau kita perhatikan ada beberapa blok kode yang ditulis secara berulang. Masing-masing aplikasi calculator, baik yang berbasis console ataupun windows form mempunyai logik aplikasi sendiri-sendiri untuk melakukan proses *Penambahan, Pengurangan, Perkalian dan Pembagian*, tentu saja hal ini tidak efisien karena salah satu tujuan OOP adalah reusability, penggunaan ulang kembali.

Ketika menemukan blok kode yang berulang, maka sebaiknya dilakukan *refactoring*. Apa itu *refactoring*? *Refactoring* adalah aktivitas mengubah kode program/struktur aplikasi tanpa menambahkan fungsionalitas yang baru. Perubahan hanya bertujuan untuk efisiensi dan efektifitas penulisan kode dengan cara menghilangkan atau menata kembali duplikasi kode yang mungkin terjadi. Ingat bagi seorang programmer, kode yang ditulis tidak hanya harus menyelesaikan masalah (solve the problem) namun penyelesaian tersebut juga haruslah indah (elegant).

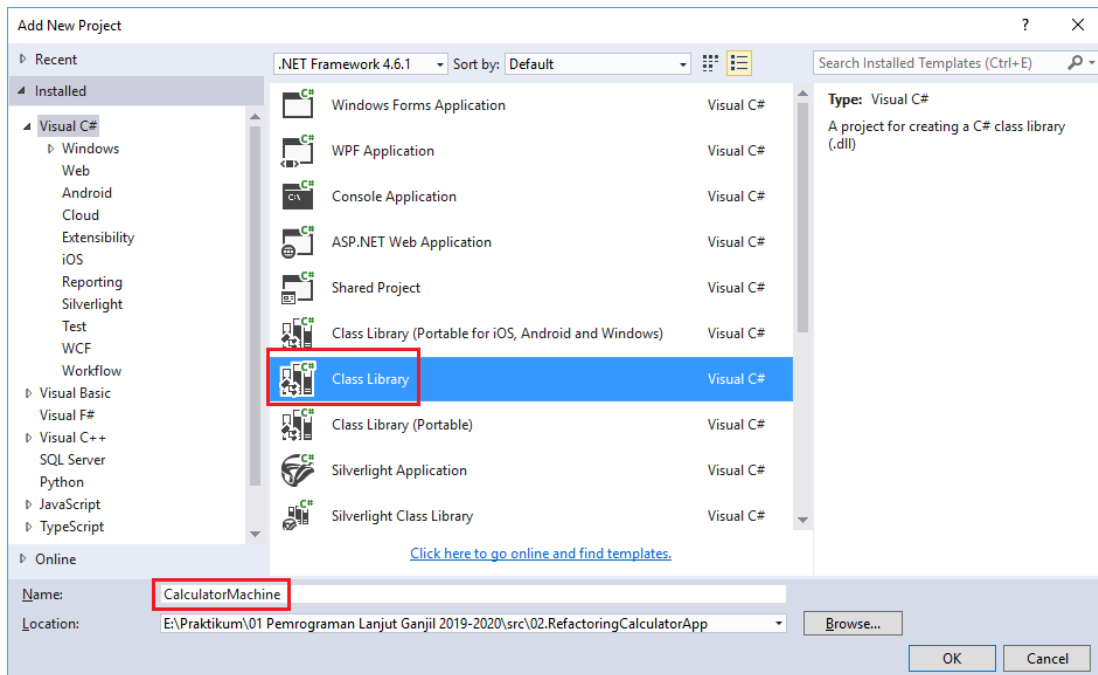
Salah satu cara untuk melakukan refactoring pada kasus duplikasi kode aplikasi calculator ini adalah dengan menambahkan project baru dengan tipe *Class Library*.

Langkah-langkah Menambahkan Project Class Library

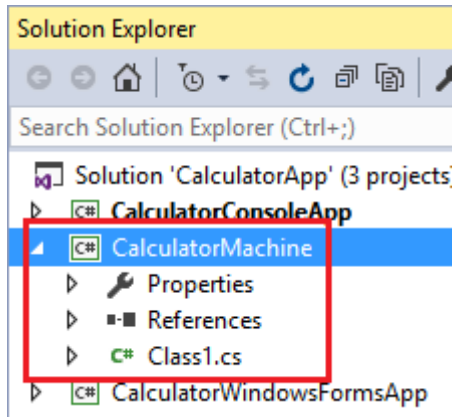
1. Aktifkan kembali project Calculator Anda di pertemuan sebelumnya. Bagi Anda yang belum menyelesaikan latihan praktikum sebelumnya, bisa menggunakan project Calculator yang ada di folder CalculatorApp
2. Masih di solution yang sama, untuk menambahkan project baru dengan cara klik kanan nama *Solution* -> *Add* -> *New Project*.



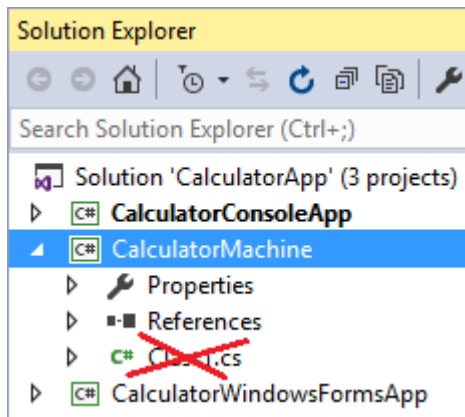
Kemudian pilih *Class Library*, untuk isian Name diisi dengan **CalculatorMachine** dan Location menyesuaikan.



Setelah project *Class Library* (*CalculatorMachine*) berhasil ditambahkan, kita bisa cek hasilnya di panel Solution.

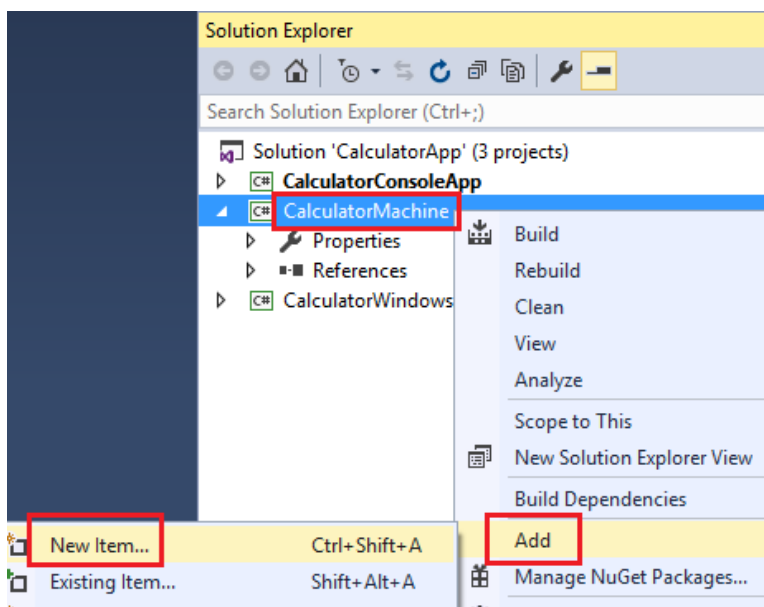


Setelah itu hapus file Class1.cs, karena untuk saat ini kita tidak membutuhkannya.

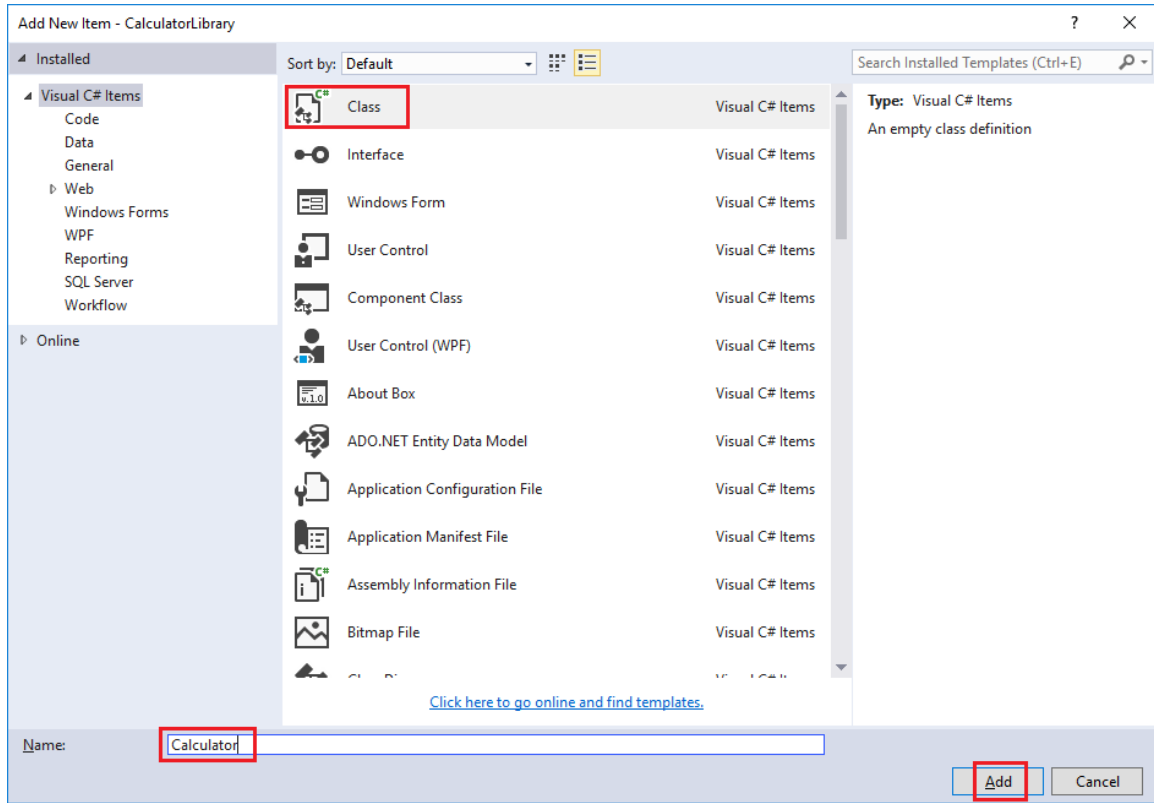


Langkah-langkah Menambahkan Class Calculator

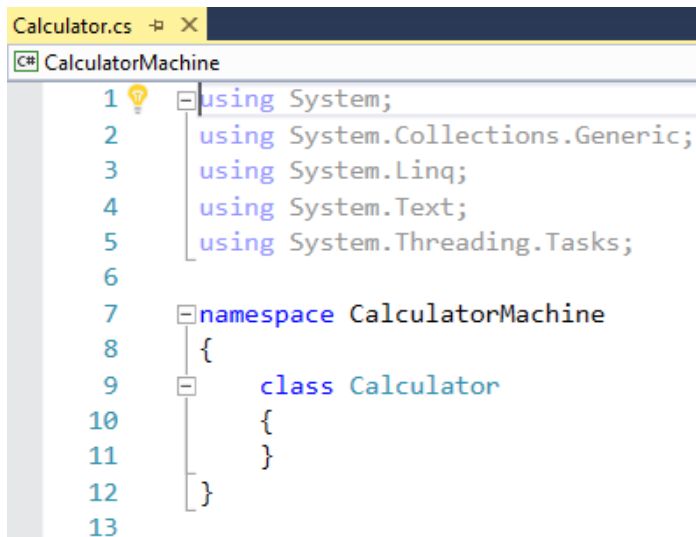
1. Klik kanan project **CalculatorMachine** -> Add -> New Item...



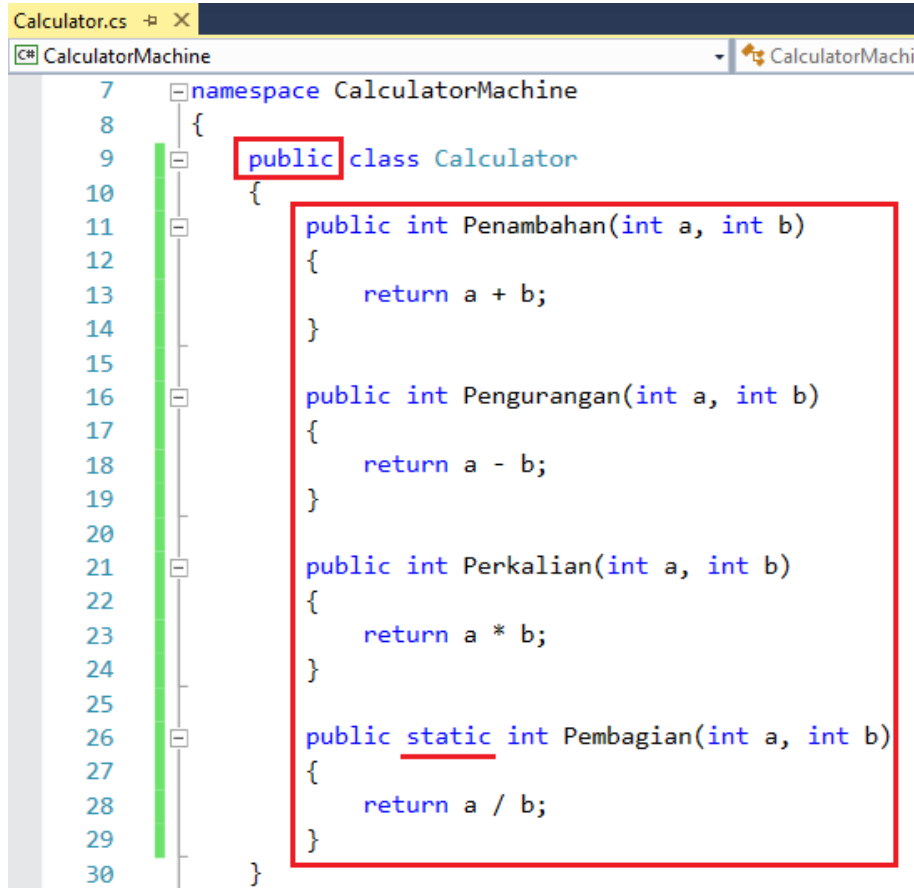
Kemudian pilih *Class*, untuk isian Name diisi dengan **Calculator**.



Setelah itu akan tampil editor code class Calculator.



Selanjutnya lengkapi kodenya seperti gambar berikut :



```
7 namespace CalculatorMachine
8 {
9     public class Calculator
10    {
11        public int Penambahan(int a, int b)
12        {
13            return a + b;
14        }
15
16        public int Pengurangan(int a, int b)
17        {
18            return a - b;
19        }
20
21        public int Perkalian(int a, int b)
22        {
23            return a * b;
24        }
25
26        public static int Pembagian(int a, int b)
27        {
28            return a / b;
29        }
30    }
```

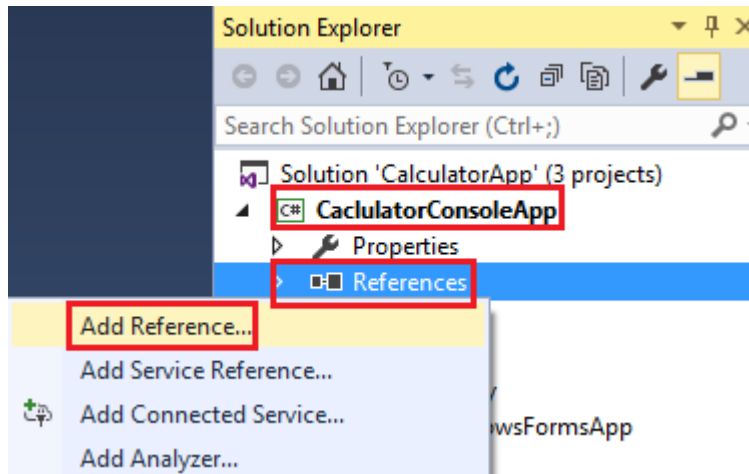
Pada baris ke 9, kita menambahkan access modifier `public` untuk deklarasi class `Calculator`, tujuannya adalah agar class `Calculator` ini bisa diakses lintas project/assembly. Untuk setiap method calculatornya juga menggunakan access modifier `public` (baris 11 – 29). Khusus untuk method `Pembagian` ada tambahan keyword `static` yang artinya method `Pembagian` ini adalah miliknya class sehingga hanya bisa dipanggil langsung dari class bukan dari objeknya.

Sampai tahap ini project `CalculatorMachine` sudah selesai, gampang bukan? 😊

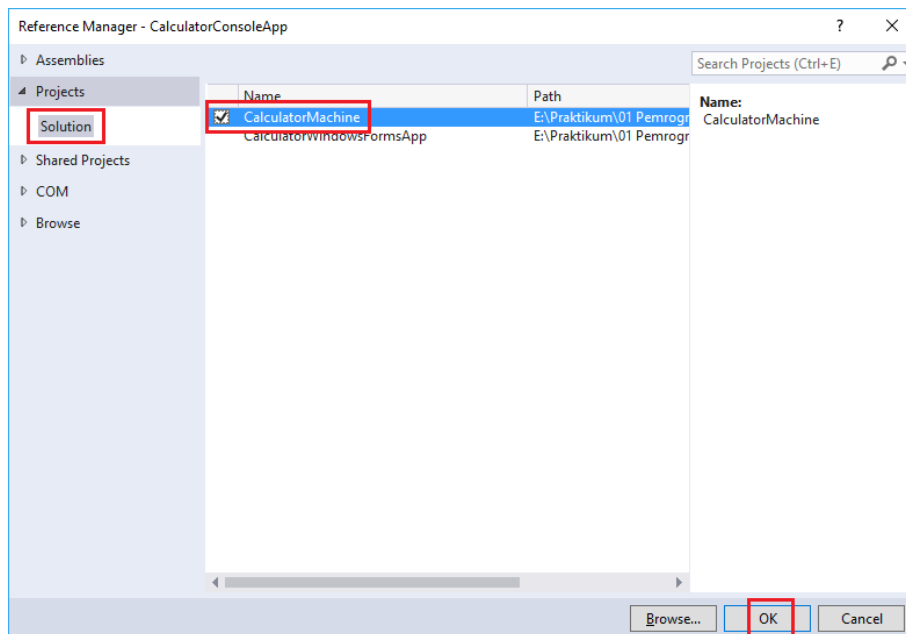
Langkah-langkah Menambahkan Referensi Project `CalculatorMachine`

Setelah pembuatan project `CalculatorMachine` selesai, langkah berikutnya adalah menambahkan referensi project `CalculatorMachine` ke project `CaclulatorConsoleApp`. Adapun tujuannya adalah agar project `CaclulatorConsoleApp` bisa mengakses semua method `public` yang ada pada project `CalculatorMachine`. Berikut langkah-langkahnya:

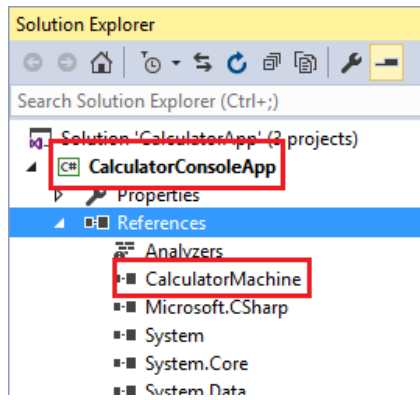
1. Aktifkan project `CaclulatorConsoleApp`, kemudian klik kanan node `References` -> `Add Reference ...`



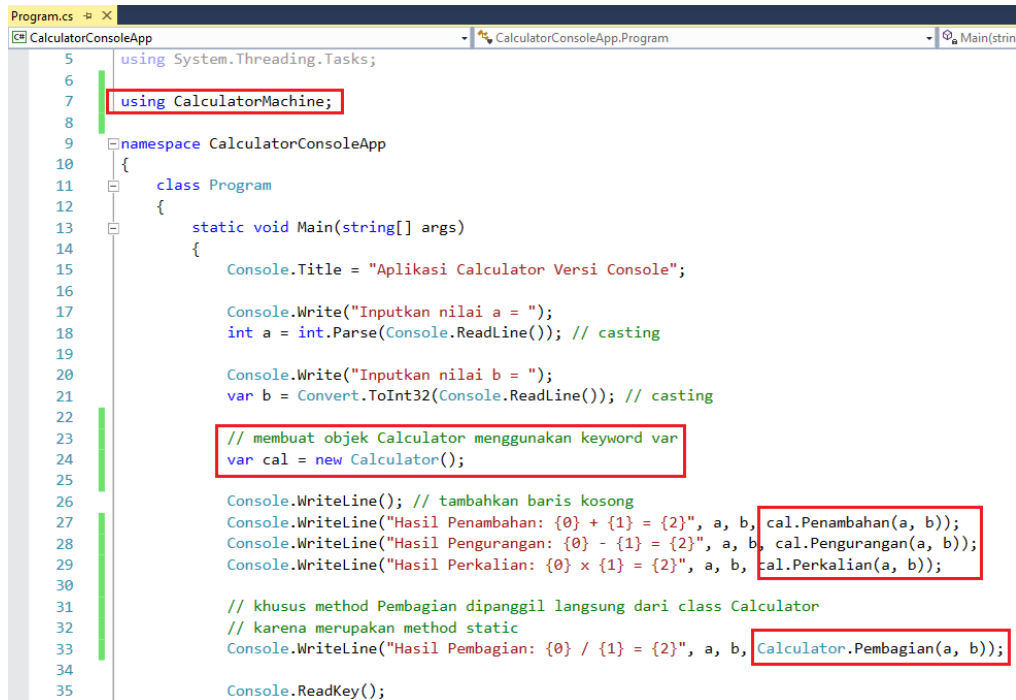
Aktifkan panel Solution, kemudian pilih CalculatorMachine



Jika berhasil project CalculatorMachine akan terdaftar di node References.



2. Selanjutnya kita mulai me-*refactoring* kode dari class Program yang ada di project CaclulatorConsoleApp



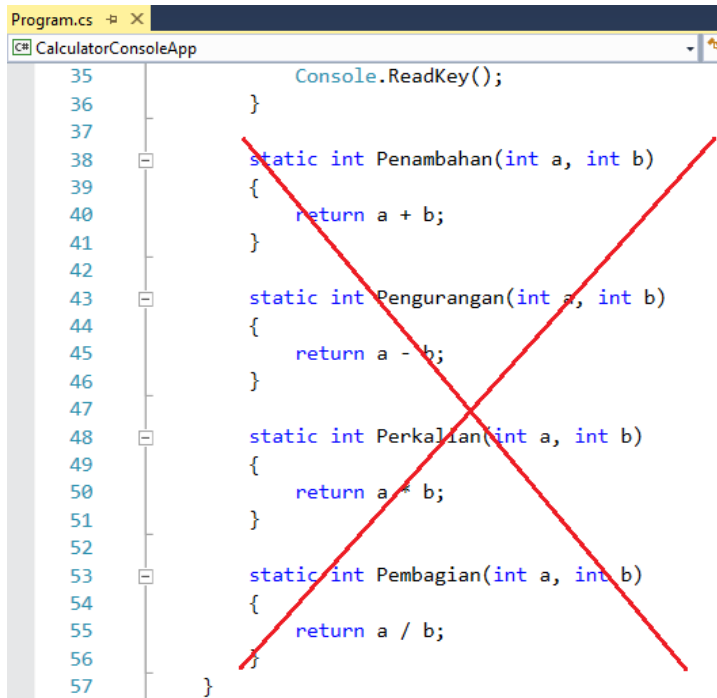
```
5 using System.Threading.Tasks;
6
7 using CalculatorMachine;
8
9 namespace CalculatorConsoleApp
10 {
11     class Program
12     {
13         static void Main(string[] args)
14         {
15             Console.Title = "Aplikasi Calculator Versi Console";
16
17             Console.WriteLine("Inputkan nilai a = ");
18             int a = int.Parse(Console.ReadLine()); // casting
19
20             Console.WriteLine("Inputkan nilai b = ");
21             var b = Convert.ToInt32(Console.ReadLine()); // casting
22
23             // membuat objek Calculator menggunakan keyword var
24             var cal = new Calculator();
25
26             Console.WriteLine(); // tambahkan baris kosong
27             Console.WriteLine("Hasil Penambahan: {0} + {1} = {2}", a, b, cal.Penambahan(a, b));
28             Console.WriteLine("Hasil Pengurangan: {0} - {1} = {2}", a, b, cal.Pengurangan(a, b));
29             Console.WriteLine("Hasil Perkalian: {0} x {1} = {2}", a, b, cal.Perkalian(a, b));
30
31             // khusus method Pembagian dipanggil langsung dari class Calculator
32             // karena merupakan method static
33             Console.WriteLine("Hasil Pembagian: {0} / {1} = {2}", a, b, Calculator.Pembagian(a, b));
34
35             Console.ReadKey();
36         }
37     }
38 }
```

Pada baris ke 7, kita menambahkan perintah *using* yang diikuti nama *namespace* yaitu CalculatorMachine. Tujuannya adalah agar kita bisa langsung mengakses class Calculator tanpa perlu menyebutkan nama namespacesnya.

Kemudian di baris ke 24, kita membuat instance/objek dari class Calculator dengan nama *cal*. Di baris ke 27 – 29, kita memanggil method Calculator langsung di objek calculator dengan menggunakan format: *namaObject.NamaMethod(parameter)*. Contoh: *cal.Penambahan(a, b)*. Khusus di baris ke 33 kita memanggil method Pembagian langsung dari class Calculator, karena method Pembagian dideklarasikan dengan tambahan keyword *static*, yang artinya method ini adalah methodnya class Calculator.

```
public static int Pembagian(int a, int b)
{
    return a / b;
}
```

Kemudian hapus method *Penambahan*, *Pengurangan*, *Perkalian* dan *Pembagian* yang ada di bawah method Main.



```
35 Console.ReadKey();
36 }
37
38 static int Penambahan(int a, int b)
39 {
40     return a + b;
41 }
42
43 static int Pengurangan(int a, int b)
44 {
45     return a - b;
46 }
47
48 static int Perkalian(int a, int b)
49 {
50     return a * b;
51 }
52
53 static int Pembagian(int a, int b)
54 {
55     return a / b;
56 }
57 }
```

3. Setelah itu jalankan aplikasi dengan menekan tombol F5

Tugas 1 – Kerjakan di lab

Lakukan juga proses *refactoring* untuk mereferensikan project **CalculatorMachine** ke project **CalculatorWindowsFormsApp**.

Tugas 2 – Kerjakan di lab

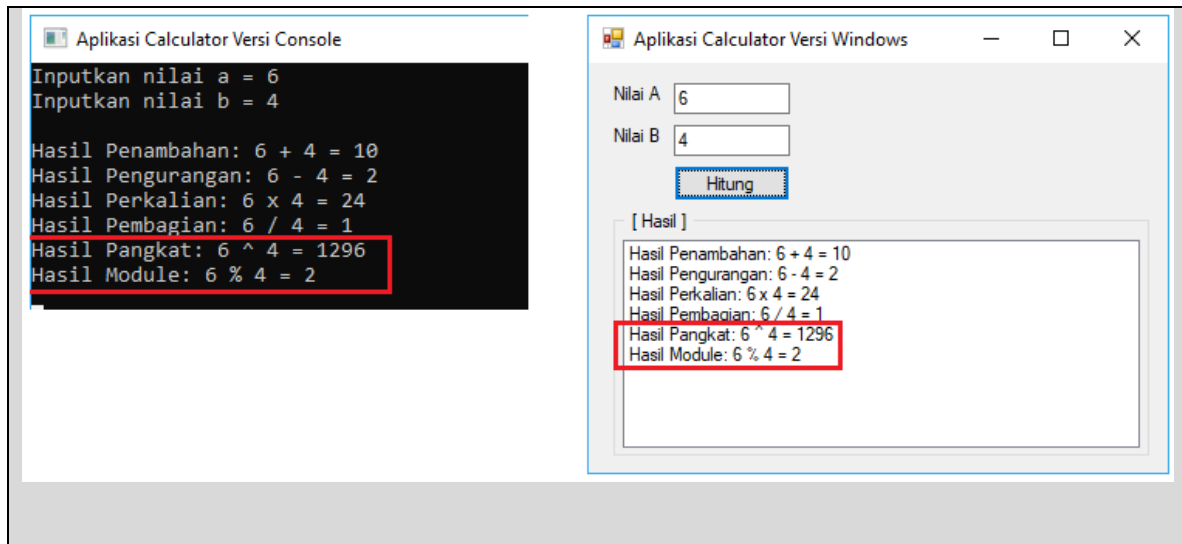
Tambahkan 2 method baru di project **CalculatorMachine** untuk menghitung *Pangkat* dan *Modulo*.

```
public int Pangkat(int a, int b)
{
    return (int)Math.Pow(a, b);
}

public static int Modulo(int a, int b)
{
    return a % b;
}
```

Kemudian tambahkan pemanggil 2 method tersebut di aplikasi **CalculatorConsoleApp** dan **CalculatorWindowsFormsApp**.

Sehingga hasil akhir aplikasi calculatornya seperti berikut:



Selesai ☺

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