**Episode 4: Method Design**

**Access modifier**: menandakan apakah class/properties tsb dapat atau tidak dapat diakses

* Public: dapat diakses siapa saja
* Private: tdk dapat diakses
* Protected: hanya dapat diakses oleh turunannya
* Internal: hanya dapat diakses oleh namespace yang sama

**Class memiliki:**

* Fields: variable yg ada dalam class
* Properties: method yang menencapsulate sebuah field
* Constructor: method yang memiliki nama sama dgn class
* Method: fungsi yg ada didalam class
  + Procedure: method yg tidak mengembalikan sebuah nilai (void)
  + Function: method yg mengembalikan sebuah nilai
  + Static Method: method yg melekat pada class

**Parameters:**

Sebuah value/nilai yang di-pass ke sebuah method, ada 3 cara untuk mengoper parameter, yaitu dgn pass by ...

* Value: value yg di pass akan di copy di dalam method, sehingga value diluar method tdk akan berpengaruh
* Reference: mengoper alokasi memori, sehingga value diluar method ikut terpengaruh
* Output: value disini tidak di pass-in ke method, namun di-pass keluar sebagai output dari method tsb. seperti ref, namun value nya tidak di-pass samsek.

Cara penggunakan ke-3 jenis parameter tsb:

int y = 10;

int x;

Test.func1(y);

// nilai y tidak terpengaruh

Test.func2(ref y);

// nilai y bertambah 1

Test.func3(out x);

// nilai x di assign menjadi 1

public class Test()

{

static void func1(int x)

{

x++;

},

static void func2(ref int x)

{

x++;

}

static void func3(out int x)

{

x = 1;

}

}

* Variable-Length Parameter List: ialah parameter yg jumlahnya tidak pasti, dan akan di pass dalam sebuah array/list.

contoh:

int y = 10;

int x = 20;

Test.func(x, y);

// memprint 30

Test.func(1,2,3,4,5)

// memprint 15

note: yg terjadi adalah semua parameter yg dipass dijadikan array, missal func(1,2,3,4,5). maka dalam method, x adalah [1,2,3,4,5]

public class Test()

{

static void func(params int[] x)

{

int sum = 0;

for (int i = 0; i < x.Length; i++)

{

sum += x[i];

}

Console.WriteLine(sum);

}

}

note: variable-length parameter list harus diletakkan di akhir apabila fungsi memiliki parameter yg lain, ini agar program tidak kebingungan.

contoh: func(int x, ref int y, params int[] z) <- benar

func(int x, params int[] z, ref int y) <- salah

**Polymorphism**

terdiri dari static dan dynamic polymorphism.

* STATIC POLYMOTRPHISM

berkaitan dgn function dan operator overloading

* function overloading: membuat function dgn nama yg sama, namun parameter atau return functionnya dapat berbeda.

class A

{

public void greet()

{

Console.WriteLine(“Hello”)

}

public void greet(string word)

{

Console.WriteLine(“Hello ” + name);

}

}

A a = new A();

a.greet()

// prints “Hello”

a.greet(“Azka”)

// prints “Hello Azka”

* operator overloading: membuat operator seperti + - dll dapat menghandle tipe data tertentu.
* DYNAMIC POLYMORPHISM

berkaitan dengan abstract class dan virtual function

* virtual function: method yang bisa di gunakan/di implementasi ulang di inherited class.

A a = new A();

B b = new B();

a.greet() // print “Hello”

b.greet() // print “Guten Tag”

class A

{

public virtual void greet()

{

Console.WriteLine(“Hello”);

}

}

class B: A

{

public override void greet()

{

Console.WriteLine(“Guten Tag”);

}

}

Class

Sebuah syntax yg mendeksripsikan atribut dan *behavior.*

Sebuah stuktur data yang terdiri data dan functionnya.

Object

Ialah instansi dari class, memiliki:

* Identity : object memiliki ID unik (nama variable objek)
* Beahavior : object dpt melakukan sesuatu (method)
* State : objek menyimpan infomasi (attribute)

**CLASS CATEGORIES**

* Instanced Class (object)

Ialah class biasa.\_.

class Hello

{

public void greet()

{

Console.WriteLine(“Hello”);

}

}

* Static Class

Class yg tidak memiliki state / class yg tdk bisa diinstantiasi

static class Hello

{

public static void greet()

{

Console.WriteLine(“Hello”);

}

}

Apabila class di declare sebagai static, maka semua membernya juga harus static. Apabila class tidak static, tetap dapat memiliki static member.

What is static?

Maksud dari static adalah membernya menempel lgsg kedalam class. Sehingga pemanggilan member tsb sedikit berbeda

class Test

{

public string a;

public static string b;

}

Test obj = new Test()

Disini, untuk memanggil member a adalah dengan obj.a, namun untuk memanggil member b, adalah dengan Test.b (menggunakan nama class) karena b adalah static member.

* Abstract Class

Class yg berfungsi hanya sebagai “template” untuk class lain, ini berkaitan dgn konsep abstraksi dan virtual function.

* Sealed Class

Class yg tidak bisa di-inherit