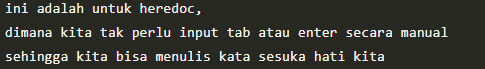
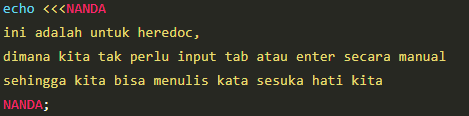
* **HEREDOC & NOWDOC**

**HEREDOC**

* Ini untuk membuat string yang panjang, sehingga tidak perlu manual untuk enter, tab dan yang lain



**NOWDOC**

* Mirip seperti heredoc, bedanya nowdoc **tidak punya kemampuan parsing atau double quote**



* **CONSTANT**

**DEFINISI**

* Bisa dibuat dengan **define(“nama\_const”, “isi\_const”);**





* **NULL**

**DEFINISI**

* Ini untuk mengisi variable dengan nilai yang tidak ada apa-apa
* Untuk mengecek, apakah variable bernilai null, dengan cara **is\_null($nama\_var)**;, outoputnya adalah **boolean**

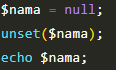




* **UNSET**

**DEFINISI**

* Ini untuk menghapus sebuah variable, dan tidak bisa mengakses variabelnya lagi
* Dengan cara **unset($nama\_var)**

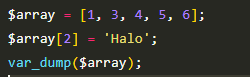


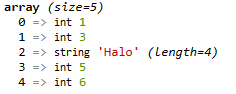


* **OPERASI ARRAY**

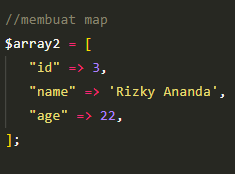
**DEFINISI**

* **$array[index]**, untuk **mengakses** data di array dengan nomor index
* **$array[index] = value**, untuk **mengubah** data di array dengan nomor index dengan value baru
* **$array[] = value**, **menambah** data di array pada posisi yang **paling belakang**
* **unset($array[index])**, **menghapus** data di array dan **index hilang** juga
* **count($array)**, untuk **mengambil total** data di array





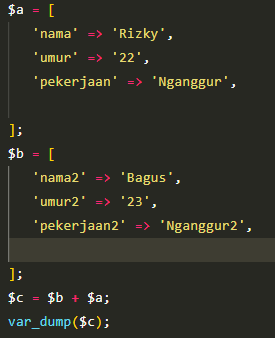
//ini adalah MAP

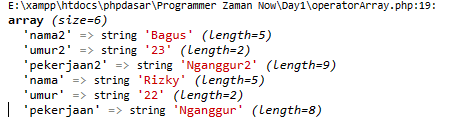


* **OPERATOR ARRAY**

**DEFINISI**

* **$a + $b** , **menggabungkan** array $a dan $b
* **$a == $b** , **true** jika key dan valuenya sama
* **$a === $b** , **true** jika key dan valuenya sama dan posisi sama
* **$a != $b** , true jika $a dan $b tidak sama
* **$a !== $b** , true jika $a dan $b tidak tidak identik





* **KONVERSI**

**DEFINISI**

* **$valueString = (string)100;,** ini akan mengubah integer menjadi string. Kita bisa mengubah menjadi float dan int dengan mengubah yang ada di dalam ( ).

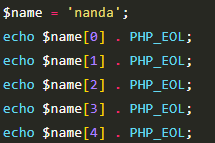




* **SIFAT STRING MENJADI ARRAY**

**DEFINISI**

* Pada php, string itu bersifat seperti array contohnya seperti berikut:

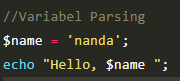




* **VARIABEL PARSING**

**DEFINISI**

* Ini adalah, kita bisa menggabungkan string dengan variable tanpa harus menggunakan titik. Contohnya





* **TERNARY OPERATOR**

**DEFINISI**

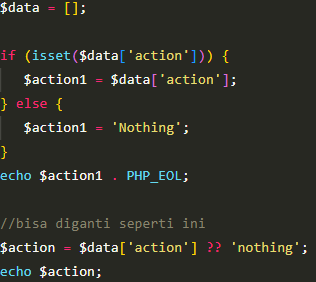
* Ini untuk **mempersingkat if and else** operator



* **COALESCING OPERATOR**

**DEFINISI**

* Ini untuk **mempersingkat** penggunaan **isset**

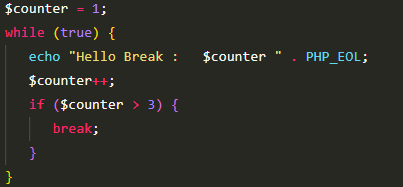


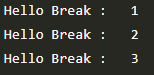


* **BREAK & CONTINUE**

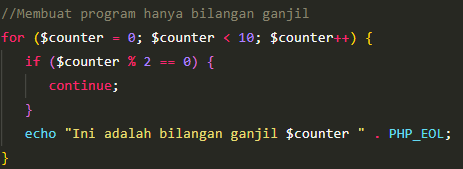
**DEFINISI**

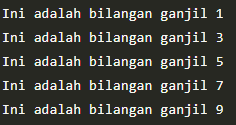
* **Break**, ini untuk **menghentikan** perulangan
* **Continue,** ini untuk **mengskip** perulangan saat ini





**//CONTINUE**

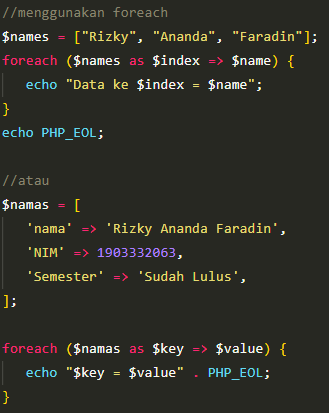


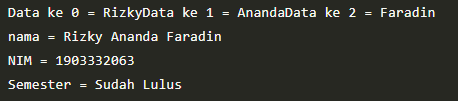


* **FOREACH ARRAY**

**DEFINISI**

* Ini untuk mengambil nilai array dengan foreach





* **DEFAULT PARAMETER**

**DEFINISI**

* Ini untuk memberikan default parameter ketika user tidak memasukkan parameter

//ini tidak boleh, usahakan berikan default parameter di belakang



* **VALID TYPES (TIPE YANG ADA DI DALAM PARAMETER)**

**DEFINISI**

* Ini untuk memberikan tipe data di parameter function, secara default php bisa mengonversi otomatis tipe data, contohnya ketika kita memasukkan argumen int, tapi tipe data dari parameter di function adalah string maka akan otomatis di konversi.
* Beberapa tipe parameter, yaitu

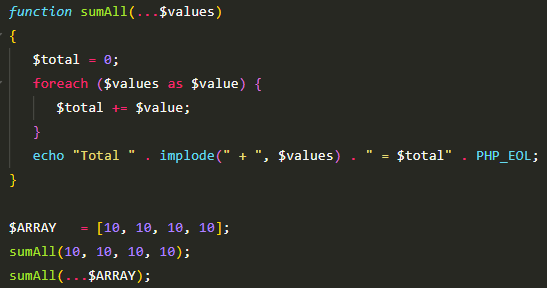
1. **Class / interface** > Parameter harus tipe **class/interface**
2. **Self** > Parameter harus sama dengan **class**, dimana function ini dibuat
3. **Array** > Parameter harus **array**
4. **Callable** > Parameter harus **callable**
5. **Bool** > Parameter harus **Boolean**
6. **Float** > Parameter harus **float**
7. **Int >** Paremeter harus **int**
8. **String >** Parameter harus **String**
9. **Interable >** Parameter harus **array** atau **Tipe Traversable**
10. **Object >** Parameter harus **object**



* **VARIABEL – LENGTH ARGUMENT LIST**

**DEFINISI**

* Ini untuk mengizinkan parameter bisa menerima banyak value.
* Ini akan otomatis membuat argument menjadi array

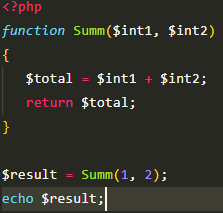




* **FUNCTION RETURN VALUE**

**DEFINISI**

* Secara default function tidak mengembalikan apapun, untuk itu gunakan return
* Kita juga bisa menambahkan **return default**, sehingga kita bisa tau tipe data kembalian dari sebuah function.



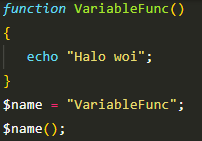
//default return



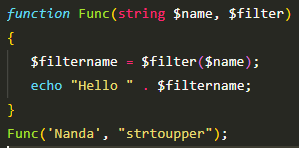
* **VARIABLE FUNCTION**

**DEFINISI**

* Kita bisa membuat function menjadi variable, dengan cara di bawah ini :



Fungsinya untuk memberikan argument berupa function

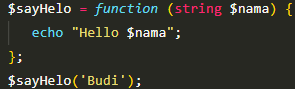


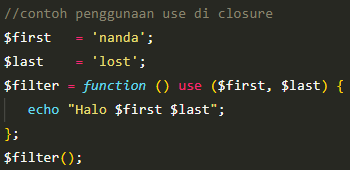


* **Anonymous Function (Closure)**

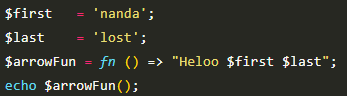
**DEFINISI**

* Kita bisa membuat function menjadi variable, dengan cara di bawah ini :
* **Secara default, kita tidak bisa mengakses variable diluar closure**, untuk mengakalinya dengan cara menggunakan **use lalu diikuti nama variable.**

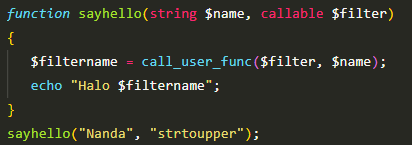




//menggunakan arrow function, harus sederhana functionnya dan kita bisa mengakses diluar closure tanpa harus menggunakan use



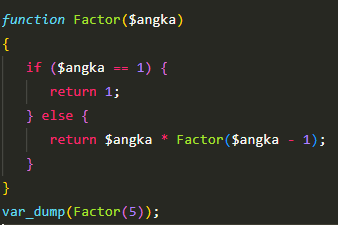
//Kita juga bisa menggunakan callback function, dengan cara seperti ini, hal ini mirip dengan arrow func



* **Recursive Function**

**DEFINISI**

* Kita bisa membuat function memanggil dirinya sendiri, contoh program factorial:



* **STRING FUNCTION**

**DEFINISI**

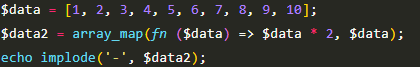
* Ada beberapa string function yaitu

1. **Join() / Implode(),** menggabungkan array menjadi string
2. **Explode(),** memecah string menjadi array
3. **Strtolower(),** mengubah string menjadi lowercase
4. **Strtoupper(),** mengubah string menjadi uppercase
5. **Substrt(),** mengambil sebagian string
6. **Trim(),** menghapus whitespace di depan dan belakang string





* **ARRAY FUNCTION**



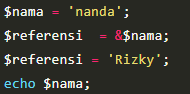


**DEFINISI**

* Ada beberapa string function yaitu:

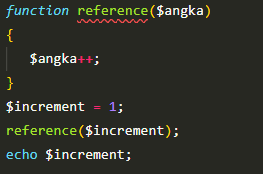
1. **Array\_keys(),** mengambil semua keys milik array
2. **Array\_values(),** mengambil semua values milik array
3. **Array\_map(),** mengubah semua data array dengan callback
4. **Sort(),** mengurutkan array
5. **Rsort(),** mengurutkan array terbalik
6. **Shuffle(),** mengubah posisi data di array secara random

* **REFERENCE**





**Tidak menggunakan reference**

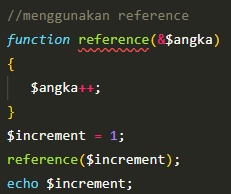




**DEFINISI**

* Saat kita mengubah isi value dari reference, maka otomatis value variable aslinya berubah, untuk membuat reference terhadap variable, bisa menggunakan &

**Menggunakan reference**



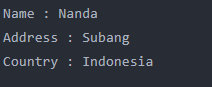


* **MANIPULASI PROPERTIES**

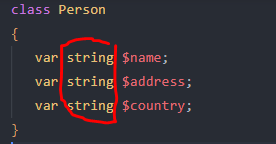
**DEFINISI**

* Kita bisa memanipulasi properties pada class, yaitu menambah dan menampilkan properties object

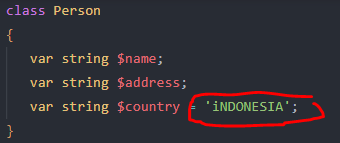




**//KITA JUGA BISA MENAMBAHKAN TIPE DATA PADA PROPERTIES**



**//KITA JUGA BISA MENAMBAHKAN DEFAULT VALUE**



* **CONSTANT**

**DEFINISI**

* Variabel yang tidak berubah2 nilainya

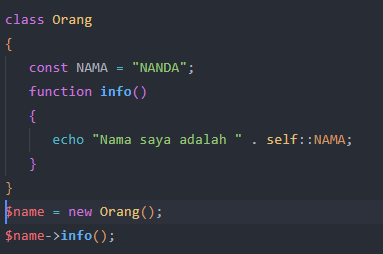
**//CARA MEMBUAT CONSTANT**



**//CARA MENGAKSES CONSTANT**



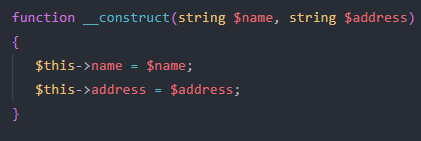
**//UNTUK MENGAKSES CONST DI DALAM KELAS YANG SAMA**



* **\_\_CONSTRACT**

**DEFINISI**

* Variabel yang tidak berubah2 nilainya



**//SETIAP KITA MENAMBAHKAN CONSTRACT DENGAN PARAMETER, KITA HARUS MENGISI ARGUMENT KETIKA INGIN MEMBUAT INSTANCE CLASS TERSEBUT**

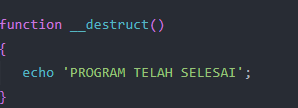


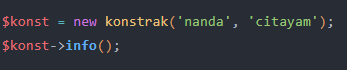
* **\_\_DESTRUCT**

**DEFINISI**

* + Ini adalah fungsi yang akan dipanggil ketika **obj dihapus dari memory**
  + Kita **tak bisa menambahkan parameter** di function destruct
  + Ini **cocok untuk menutup koneksi** ke database atau proses menulis ke file, sehingga tidak terjadi memory leak

//**KETIKA OBJ SELESAI DI PANGGIL, MAKA DESTRUCT AKAN DIEKSEKUSI**







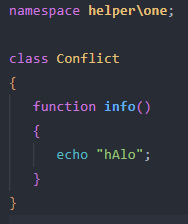
**//VOID BERARTI MENANDAI SEBUAH FUNC TIDAK MENGEMBALIKAN NILAI APAPUN**



* **NAMESPACE**

**DEFINISI**

* + Ini ketika kita ingin membuat class yang namanya sama, kita bisa manambahkan namespace untuk kode uniqenya





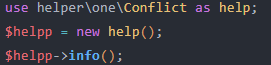
**//INI UNTUK MENUNJUKKAN BAHWA KITA MEMAKAI NAMESPACE YANG INI**



* **ALIAS**

**DEFINISI**

* + Alias adalah kemampuan membuat nala lain dari class, function atau constant yang ada



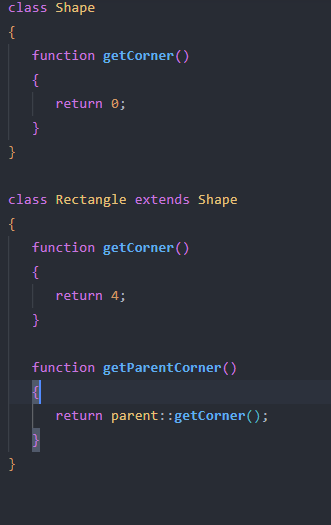
**//BISA LANGSUNG IMPORT BANYAK CLASS DI NAMESPACE YANG SAMA**



* **OVERRIDING**

**DEFINISI**

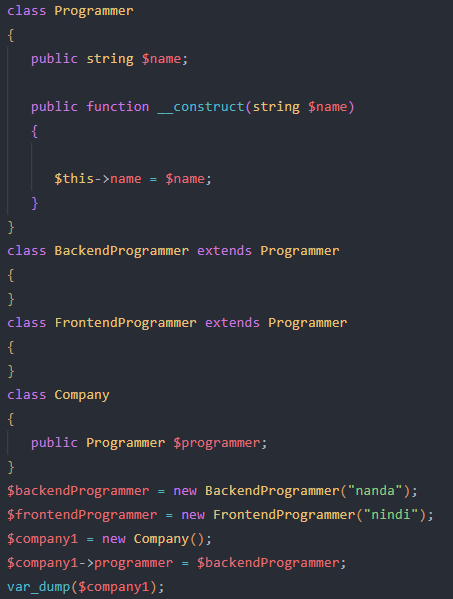
* + Ketika terdapat function yang sama antara class parent dan class child kita bisa **menggunakan keyword Parent::**

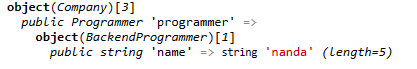


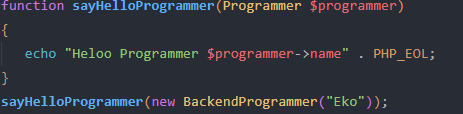
* **POLYMORPHISM**

**DEFINISI**

* + Polymorphism adalah kemampuan sebuah object berubah bentuk menjadi bentuk lain









* **INSTANCEOF**

**DEFINISI**

* + Untuk melakukan pengecekan tipe data object





* **ENCAPSULATION**

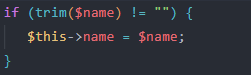
**DEFINISI**

* + Memastikan data sensitive sebuah object tersembunyi dari akses luar.
  + Membuat properties menggunakan access modifier private, sehingga tidak bisa diakses diluar





**//UNTUK MEMVALIDASI INPUTAN DARI USER**



* **INTERFACE**

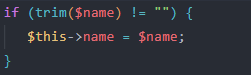
**DEFINISI**

* + Ini mirip seperti abstract, tapi ini lebih benar dari pada abstract, di interface **tidak bole punya properties, hanya const**.
  + Bisa **implements lebih dari satu interface**

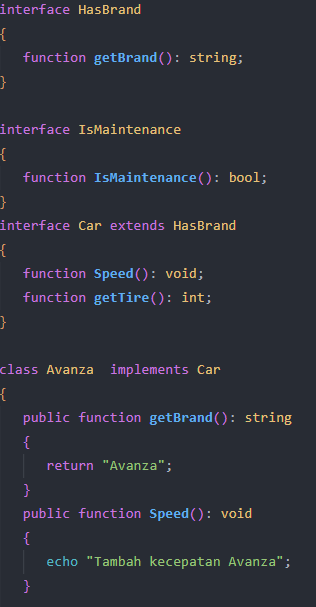




**//UNTUK MEMVALIDASI INPUTAN DARI USER**



**//INTERFACE EXTENDS INTERFACE BISA, CONTOHNYA BEGINI**



* **TRAIT**

**DEFINISI**

* + Trait **mirip dengan abstract**, tapi kita bisa tambahkan trait ke dalam class **lebih dari satu.**
  + Trait digunakan untuk **menyimpan function-function yang bisa digunakan ulang** di beberapa class.
  + Untuk menggunakan trait di class, bisa menggunakan **use**