**#8 OOP DASAR pada PHP Overriding**

Istilah overriding kita bisa membuat method di class child yg memilik nama yg sama di class parent nya, mengambil alih atau menimpa method yg kita buat di parent class nya. Kita melakukan ini karena inngin melakukan fungsionalitas yg sama tapi dengan beberapa perbedaan yg berbeda.

Lebih jelas nya kita lanjutkan studi kasus yg sebelum nya, getInfoProduk() merupakan method yg sama dengan class parent nya, tapi method di parent nya ga selengkap yg kita punya di method child nya.

Dan misalkan kalo produk nya belum ada kategori bisa instace nya class Produk, nah cara menjalankan Overriding di class child nya gini :

class Komik extends Produk {

    public function getInfoLengkap() {

        $str = "Komik : " . parent::getInfoLengkap() . " - {**$this**->jmlHalaman} Halaman";

        return $str;

    }

}

class Game extends Produk {

    public function getInfoLengkap() {

        $str = "Game : ". parent::getInfoLengkap() . " ~ {**$this**->waktuMain} Jam";

        return $str;

    }

}

Nah keyword (parent:: ) itu untuk memanggil method yg sama di parent nya.

**Overriding ke-2**

Saat kita belajar inheritance kan sebetulnya akan lebih baik variable spesial itu di masing-masing fungsi misal $jmlHalaman itu di Class Komik karena kan spesifikasi nya dan variable $waktumain itu di Class Game ya.

Kita coba untuk yg $jmlHalaman dulu aja ya, variable $jmlHalaman di apus aja di \_\_constractor nya ya, nah berarti ke apus tuh ya si jmlHalaman nya berarti kita harus buat constractor baru lagi di Class Komik.

Misalkan saat memanggil atau instasiasi class Komik kan kita kirim parameter ke Class Komik ya ke method constract itu, Nah sebelum nya kita belum buat fungsi constract nya maka dia akan otomatis nyari ke parent nya.

Overriding :

Nah kalo kita buat constructor nya maka kita lakukan juga overriding di function atau method tersebut ya gini :

$produk2 = new Komik("Twenty Five Twenty One", "Jeong Jiman", "TVN", 1500000, 100, 0);

$produk3 = new Game("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 0, 50);

Itu saat mengirim kan ke parameter sebelum constractor Class komik di buat dan ini sekarang :

class Komik extends Produk {

    public $jmlHalaman;

    public function \_\_construct($judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0, $jmlHalaman = 0) {

        parent::\_\_construct($judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0);

        this->jmlHalaman = $jmlHalaman;

    }

    public function getInfoLengkap() {

        $str = "Komik : " . parent::getInfoLengkap() . " - {**$this**->jmlHalaman} Halaman";

        return $str;

    }

}

$produk2 = new Komik("Twenty Five Twenty One", "Jeong Jiman", "TVN", 1500000, 100);

$produk3 = new Game("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 50);

Jadi ya kita memanggil function \_\_constrac nya di dalam function \_\_constrac punya Class Komik dan itu kita kasih parameter tapi hanya sampe harga karena jumlah halaman nya kan udah ada di variable nya.

class Komik extends Produk {

    public $jmlHalaman;

    public function \_\_construct($judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0, $jmlHalaman = 0) {

        parent::\_\_construct($judul, $penulis , $penerbit, $harga);

**$this**->jmlHalaman = $jmlHalaman;

    }

    public function getInfoLengkap() {

        $str = "Komik : " . parent::getInfoLengkap() . " - {**$this**->jmlHalaman} Halaman";

        return $str;

    }

}

Ada perbaikan saat memangggil parent nya nilai default nya jangan di tulis ya soal nya ketimpa nilai nya. Dan untuk yg game cara kerja nya sama aja. Hasil nya sama saja seperti sebelum nya tapi kita sudah melakukan nya jauh lebih baik ya untuk jangka yg lebih besar gitu.

Selanjutnya kita belajar akses moifier Visibility, sekarang kita buat semua nya public itu salah ya kurang tepat karena ada beberapa property dan method yg gabisa di akses secara sembarangan.