**#14 OOP DASAR pada PHP Abstract Class (2)**

Kita kembali ke kasus kita sebelum nya di bagian pemabahasan setter dan getter, kita punya 4 buah class, class utama nya class Produk dan ada 2 buah class turunan nya yaitu Game dan Komik, satu nya lagi class CetakInfoProduk.

Jadi cerita nya aplikasi sederhana ini kita gunain untuk mengelola data produk yg saat ini produk nya hanya ada 2 yaitu Komik dan Game dan kita punya satu class terakhir yg digunakan untuk mencetak info lengkap dari sebuah produk.

Sebelum masuk ke abstrak kita akan perbaiki class CetakInfoProduk itu, sebelum nya saat kita mau instasiasi sebuah produk itukan cara nya kita instasiasi produk nya lalu kita cetak satu-satu.

Nah saya ingin class cetakInfoProduk ini mencetak banyak produk sekaligus, jadi ada property isi nya array yg akan menyimpan produk-produk yg akan kita cetak sekaligus. Lalu cara menabahkan produk-produk nya kedalam array itu dengan membuat method baru nanti menerima parameter hanya dari Class Produk dan turunan nya gini :

class CetakInfoProduk{

    public $daftarProduk = [];

    public function tambahProduk(Produk $produk) {

**$this**->daftarProduk[] = $produk;

    }

    public function cetak(Produk $produk) {

        return $produk->getInfoLengkap();

    }

}

Nah nantinya method itu akan menambahkan setiap ada produk baru yg terdaftar ke dalam property daftarProduk, lalu method cetak() akan kita ubah gini :

class CetakInfoProduk{

    public $daftarProduk = [];

    public function tambahProduk(Produk $produk) {

**$this**->daftarProduk[] = $produk;

    }

    public function cetak() {

       $str = "DAFTAR PRODUK : <br>";

       foreach (**$this**->daftarProduk as $p ) {

         $str .= "- {$p->getInfoLengkap()} <br>";

       }

       return $str;

    }

}

getInfoLengkap itu ada ya method nya di atas okey, jadi p nya itu kan turunan dari class produk ya kaya game dan komik gitu nah dia manggil getInfoLengkap di dalam class game atau komik gitu.

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

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

$newP = new CetakInfoProduk();

$newP->tambahProduk($produk2);

$newP->tambahProduk($produk3);

echo $newP->cetak();

nah itu cara kita pas panggil method nya gayss.

MASUK MATERI ABSTRAK :

Jadi kan kita punya class produk ya dimana produk ini adalah class utama untuk mendefinisikan komponen dasar dari sebuah produk, nah sebetul nya produk yg nantinya akan kita gunakan yg 2 itu komik dan game ya.

Jadi kita bisa liat bahwa kalo nanti kita tidak akan pernah meninstasiasi class produk itu maka class produk itu bisa kita jadikan sebagai class abstract, sekali lagi ini merupakan keputusan perancangan atau desain bukan benar salah nya ya, terserah yg buat aja ini mah.

Kalo kepetusan nya class produk tidak akan kita instansiasi bisa kita jadikan class abstract dia, kalo kalian liat disitu ada method getInfoLengkap yg nanti nya digunakan juga di class-class turunan nya kita jadikan saja dia sebagai method abstarct nya ya. Gini :

abstract class Produk{

  private $judul ,$penulis ,$penerbit, $harga,  $diskon = 0;

*// Constructor Method*

    public function \_\_construct($judul = "judul",

    $penulis = "penulis",

    $penerbit = "penerbit", $harga = 0) {

**$this**->judul = $judul;

**$this**->penulis = $penulis;

**$this**->penerbit = $penerbit;

**$this**->harga = $harga;

    }

   abstract public function getInfoLengkap();

   public function getInfo() {

    $str = "{**$this**->judul} | {**$this**->getLabel()}, (Rp. {**$this**->harga})";

        return $str;

   }

}

Jadi nya getInfoLengkap() jadi method abstract tetapi isi dari method nya kita pindahin ke getInfo() agar nanti di class turunan nya bisa tetap di panggil gini :

class Game extends Produk {

    public $waktuMain;

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

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

**$this**->waktuMain = $waktuMain;

    }

    public function getInfoLengkap() {

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

        return $str;

    }

}

Nah jadi nya kita bisa pake method abstract getInfoLengkap di komik dan game dengan syarat harus di implmentasikan ya di class produk nya kan ga di implementasikan ya tuh. Jika cara ini menjadi rumit kalian bisa gausah pake abstract karena ini hanya sebuah kasus sederhana aja.

Karena mungkin keliatan nya jadi tambah banyak code nya, ini hanya sebuah contoh saja ya agar mudah dipahami. Jadi kita melakukan hal yg sama seperti sebelum nya tapi kita menerap kan konsep abstrak, dengan melakukan konsep abstrak ini kita sudah melakukan konsep OOP nya selangkah lebih baik lagi.