PHP

Object Oriented Programming

Bangsa Cerdas Institute Batch 1 : January 27, 2017

APA ITU OBJECT?

PHP primitive examples

- **booleans**
- **floating-point numbers**
- **integers**
- **strings**

Objects

- **complex data structure**
- **attributes**
- **behaviors**
- **can contain multiple data types**

primitive vs object

Bangsa Cerdas.com

string
Jakarta

object
nomor jalan: 123
kota: Jakarta
propinsi: DKI

apa itu class?

- blueprint of **attribute** and **behaviors**
- **objects** are **instances** of a class
  - instances have attributes and behaviors of class
- classes **modularize program functions**

class: Alamat

attributes:

kota

propinsi

kodepos

behavior:

cari kode pos

instances dari **Alamat**

attributes:	**attributes:**
kota: Medan	kota: Surabaya
propinsi: Sumatera Utara	propinsi: Jawa Timur
kodepos: 15432	kodepos: 32156
behavior:	**behavior:**
cari kode pos	cari kode pos

OOP features

- **abstraction**
  - defines data and structures
  - hide implementation
- **encapsulation**
  - expose functionality, restricts access

OOP features

- **hierarchy**
  - Inherited attributes and behaviors
  - Incremental development
- **modularity**
  - functionality is broken into task specific pieces
- **polymorphism**
  - interact with classes without knowing class it is

DEFINING A CLASS

PHP class naming

- start with letter or underscore
- the remaining characters must be letters, numbers, or underscore
- no limit on lenght

class name best practice

upper camel case naming

```
<?php
class Alamat {}
class AlamatTinggal {}
class Alamat_Tinggal {}
class alamat_tinggal {}
class 21_jump_street {}
```

MENDEFINISIKAN CLASS

```php
<?php
/**
 * Class Alamat Tinggal
 */
class Alamat {
	// Alamat Jalan
	public $alamat_jalan_1;
	public $alamat_jalan_2;
	// Nama Kota
	public $nama_kota;
	// Nama Propinsi
	public $nama_propinsi;
	// Kodepos
	public $kodepos;
	// Nama Negara
	public $nama_negara;
}
```

```
/**
* Menampilkan alamat pada HTML.
* @return string
*/
function tampilkan() {
	$output = '';
	// Alamat Jalan
	$output .= $this->alamat_jalan_1;
	if($this->alamat_jalan_2) {
		$output .= '<br />'.$this->alamat_jalan_2;
	}
	// Kota, Propinsi, Kodepos
	$output .= '<br />';
	$output .= $this->nama_kota . ', ' . $this->nama_propinsi;
	$output .= ', ' . $this->kodepos;
	// Negara
	$output .= '<br />';
	$output .= $this->nama_negara;
	return $output;
}
```

INSTANTIATING OBJECT & ACCESSING ITS CONTENT

The 'new' keyword

Untuk membuat sebuah ***object*** dari ***class***, Anda perlu menggunakan kata kunci ***‘new’***

```php
<?php

require 'class.Alamat.inc';

echo '<h1>Instansiasi Alamat</h1>';
$alamat = new Alamat;

echo '<h2>Alamat Kosong</h2>';
echo '<tt><pre>' . var_export($alamat, TRUE) . '</pre></tt>';
```

```
echo '<h2>Set Alamat</h2>';
$alamat->alamat_jalan_1 = 'Jalan Putuh Raya Blok Z2';
$alamat->nama_kota = 'Jakarta Utara';
$alamat->nama_propinsi = 'DKI Jakarta';
$alamat->kodepos = '15432';
$alamat->nama_negara = 'Indonesia';
echo '<tt><pre>' . var_export($alamat, TRUE) . '</pre></tt>';


echo '<h2>Tampilkan Alamat</h2>';
echo $alamat->tampilkan();
```

Constructor Function

Constructor Function adalah fungsi yang memiliki tipe spesial, karena dia akan dipanggil secara otomatis ketika object dari suatu kelas dibuat

Konstanta Class

- Sebagaimana
  sifat *konstanta* reguler, *class constant* juga tidak bisa diubah nilainya ketika sudah didefenisikan
- Untuk membuat *class constant* di dalam PHP, kita menggunakan perintah: **const**
- Untuk mengakses nilai *konstanta*, gunakan *double colon* '**::**'

```
class laptop {
    const DOLLAR = '12000';
}
echo "Harga dollar saat ini = Rp. ".laptop::DOLLAR;


// buat objek dari class laptop (instansiasi)
$laptop_baru = new laptop();

echo "Harga dollar saat ini = Rp ".$laptop_baru::DOLLAR;
```

```
class laptop {
   // buat konstanta
   const DOLLAR = '12000';

   // buat method
   public function beli_laptop($harga) {
     return "Beli Komputer Baru, Rp. ".$harga*self::DOLLAR;
   }
}

// buat objek dari class laptop (instansiasi)
$laptop_baru=new laptop();

echo $laptop_baru->beli_laptop(400);
```

```
class komputer {
	const DOLLAR = '11000';
}
class laptop extends komputer {
	const DOLLAR = '12000';
	public function beli_komputer($harga){
		return "Beli Komputer Baru, Rp .".$harga*parent::DOLLAR;
	}
	public function beli_laptop($harga){
	    return "Beli Komputer Baru, Rp .".$harga*self::DOLLAR;
	}
}
// buat objek dari class laptop (instansiasi)
$laptop_baru=new laptop();
echo $laptop_baru->beli_laptop(400);
echo "<br />";
echo $laptop_baru->beli_komputer(400);
```

The '$this' variable

$this adalah built-in variabel yang menunjuk ke sebuah obyek. Atau dengan kata lain, **$this** adalah variabel self-referensi khusus

Inheritance

Inheritance adalah penurunan sebuah sifat, secara teknisnya dalam pemograman adalah penurunan struktur data agar ***Class*** turunannya memiliki ***variable*** dan ***method*** yang sama

Function Overriding

Jika kita membuat method pada *child class* dengan nama yang sama seperti pada nama *method* yang ada pada *parent class*

Access Modifiers

Public	Private	Protected
Dalam PHP *Method* dan *variable* yang sudah diset menjadi *public* dapat diakses dimana saja.	Dalam PHP *Method* dan *variable* yang sudah diset menjadi *private* hanya dapat diakses oleh *Class* itu sendiri	Dalam PHP *Method* dan *variable* yang sudah diset menjadi *protected* dapat diakses oleh *Class* itu sendiri dan *Class* turunanya

Ketiga keyword tersebut digunakan didalam sebuah **Class** yang bisa kita gunakan untuk mengubah sifat akses pada suatu **variable** dan **method**

Final Method dan Final Class

- Mekanisme untuk **melarang** class anak untuk membuat method yang akan menimpa method class induk
- Atau bahkan melarang sebuah *class* untuk diturunkan sama sekali
- Gunakan keyword: **final**

```
class komputer{
   final public function lihat_spec(){
     return "Lihat Spesifikasi Komputer";
   }
}

class laptop extends komputer{
   public function lihat_spec(){
     return "Lihat Spesifikasi Laptop";
   }
}

$laptop_baru=new laptop();
```

Abstract Class dan Abstract Method

Abstract Class adalah sebuah class yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai '*kerangka dasar'* bagi class turunannya. Di dalam *abstract class* umumnya akan memiliki *abstract method*

Abstract Class dan Abstract Method

Abstract Method adalah sebuah '*method dasar*' yang harus diimplementasikan ulang di dalam class anak (*child class*). *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya '**signature'**-nya saja. **Signature** dari sebuah *method* adalah bagian method yang terdiri dari nama method dan parameternya (jika ada)

Object Interface

- **Interface** adalah sebuah '*kontrak*' atau perjanjian *implementasi method*
- Bagi *class* yang menggunakan *object interface*, harus mengimplementasikan ulang seluruh *method* yang ada di dalam *interface*
- ***Interface*** adalah implementasi method yang harus '*tersedia*' dalam sebuah objek
- ***Interface*** lebih berperan untuk *menyeragamkan method*

Polymorphism

- Konsep dimana terdapat banyak class yang memiliki signature method yang sama
- Implementasi dari method-method tersebut diserahkan kepada tiap class, akan tetapi cara pemanggilan method harus sama
- Membuat struktur pola dari class dan turunannya
- Menekankan alur kode program yang terorganisir untuk mengurangi adanya perulangan kode program

READY TO CHALLENGE?

Latihan

1. Dalam script ***'class-kendaraan.php'***, buatlah class baru bernama ***'pesawat'*** yang merupakan turunan dari class kendaraan
2. Dalam class ***'pesawat'*** yang telah dibuat, definisikan sebuah properti ***'tinggiMaks'*** dengan sifat ***private*** untuk menyatakan ketinggian maksimum pesawat dan ***'kecepatanMaks'*** dengan sifat ***private*** untuk menyatakan kecepatan maksimum pesawat
3. Dalam class ***'pesawat'***, buatlah sebuah method bernama ***setTinggiMaks()*** untuk mensetting properti ***'tinggiMaks'*** dan ***setKecepatanMaks()*** untuk setting properti ***'kecepatanMaks'*** pesawat

Latihan

4. Dalam class ***'pesawat'***, buatlah method bernama ***bacaTinggiMaks()*** untuk mengakses properti ***'tinggiMaks'***.
5. Dalam class ***'pesawat'***, buatlah method bernama ***biayaOperasional()*** untuk menentukan biaya operasional pesawat, dimana untuk menghitung biaya ini tergantung dari harga pesawat yaitu dirumuskan:
   - Jika tinggi maksimum pesawat lebih dari 5000 feet dan kecepatan maks lebih dari 800 km/jam, maka biaya operasional = 30% dari harga pesawat
   - Jika tinggi maksimum pesawat 3000-5000 feet dan kecepatan maks 500 – 800 km/jam, maka biaya operasional = 20% dari harga pesawat
   - Jika tinggi maksimum pesawat kurang dari 3000 feet dan kecepatan maks kurang dari 500 km/jam, maka biaya operasional = 10% dari harga pesawat
   - Selain itu, biaya operasionalnya = 5% dari harga pesawat

Latihan

6. Berdasarkan ketentuan pada nomor 1 s/d 5, tentukan biaya operasional dari pesawat-pesawat ini:

Merek Pesawat	Harga (juta)	Tinggi Maks (feet)	Kecept Maks (km/jam)
Boeing 737	2.000	7500	650
Boeing 747	3.500	5800	750
Cassa	750	3500	500

Contoh tampilan output yang diharapkan adalah sebagai berikut:

Biaya operasional pesawat **'Boeing 737'** dengan harga **Rp 2.000.000.000** yang memiliki tinggi maksimum **7500 feet** dan kecepatan maksimum **650 km/jam** adalah **Rp. XXXXXXX**

TERIMA KASIH