POLITEKNIK
POS INDONESIA
Logistics Expert

Pemrograman I

Design dan Pemrograman Website

http://roniandarsyah.poltekpos.ac.id/

Roni Andarsyah, ST., M.Kom

Lecture Series

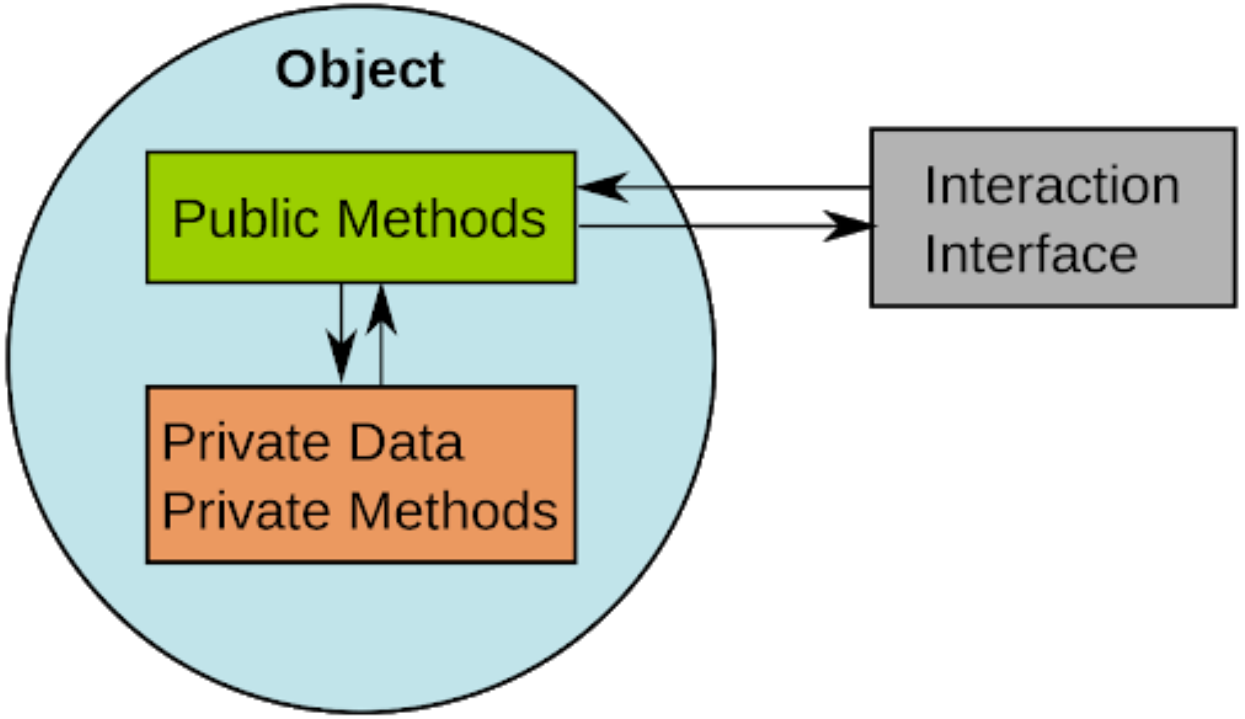
Berorientasi Object

Object Oriented Programming

- Programming Style/Paradigma Pemrograman
- Sebelum belajar OOP pastikan sudah mempelajari **PHP Prosedural**
- Menyusun semua kode program dan struktur data sebagai objek
- Objek adalah unit dasar dari program
- Objek menyimpan data dan perilaku *(behavior)* atribut dan methode
- Objek bisa saling berinteraksi
- Java, Ruby, Python, C++, Javascript, PHP5

Kelebihan Object Oriented Programming

- Representasi dunia nyata
- Enkapsulasi (aman) & Abstraksi (menyembungikan) Data
- Reusablity
- Skalabilias & Ekstensibilitas
- Kemudahan pengelolaan
- Kolaborasi
- Digunakan oleh Framework



Konsep Object Oriented Programming

Basic

- Class & Object
- Property & Method
- Construktor
- Object Type
- *Inheritance*
- *Visibility / Acces Modifier*
- *Setter & Getter*
- Static Method

Advanced

- *Abstract & Interface*
- *Interceptor (Magic Method)*
- *Object Cloning*
- *Callbacks & Closures*
- *Namespaces & Autoloading*
- ...

Class, Object, Property dan Method

Class

- Blueprint/Template untuk membuat *Instance* dari object
- Class mendefinidikan Object
- Menyimpan *data* dan *perilaku* yang disebut dengan *property* dan *Method*
- Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nantinya adalah hasil cetakan dari class, yang disebut **object.**

Property

- Object adalah hasil cetak dari class, atau hasil *'konkrit'* dari class.

Property

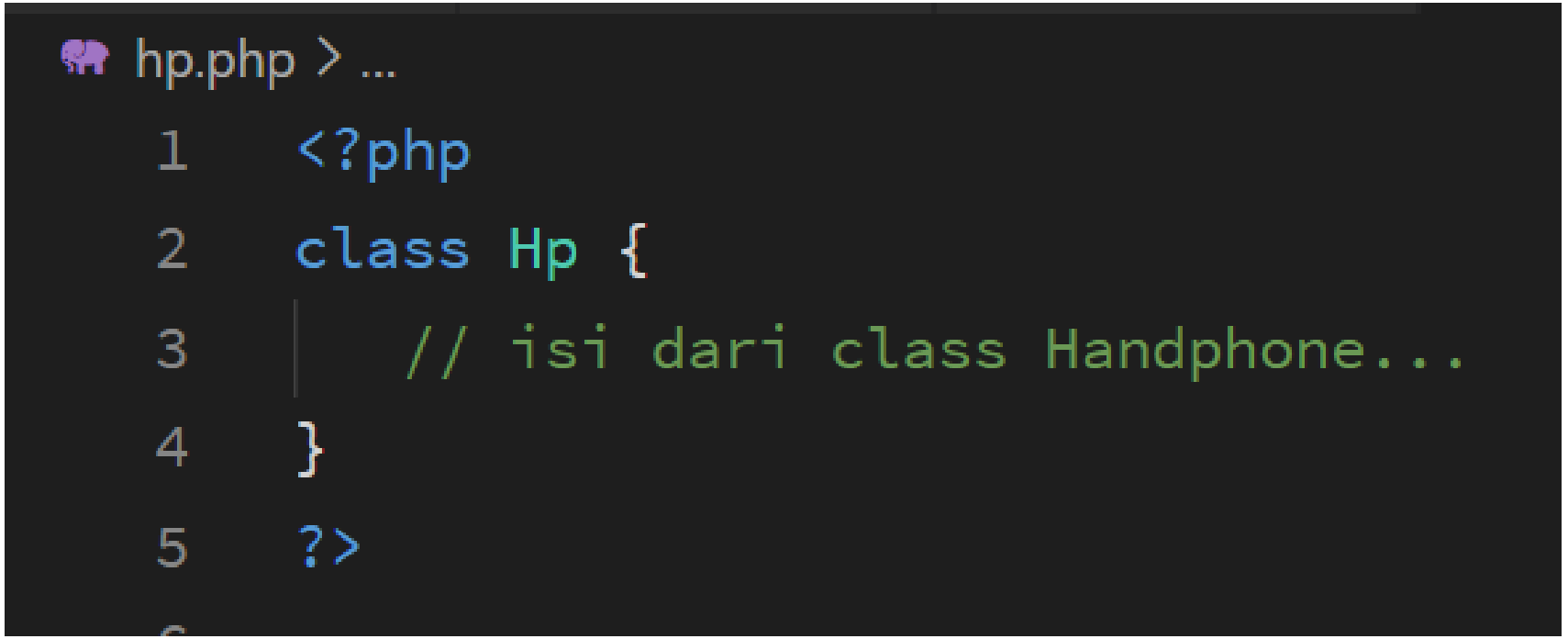
- Property (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah class.

Method

- Method adalah tindakan yang bisa dilakukan di dalam class.

Membuat Class

- Diawali dengan keyword ***class,*** diikuti nama dan dibatasi dengan **{}** untuk menyimpan property dan method
- Aturan penamaan class sama seperti variable



Property dalam Pemrograman Berbasis Objek

- **Property** (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah class.
- Melanjutkan analogi tentang HP, property dari HP bisa berupa merk, warna, ukuran layar, dan lain-lain.
- **Property** ini sebenarnya hanyalah variabel yang terletak di dalam class. Seluruh aturan dan tipe data yang biasa diinput ke dalam variabel, bisa juga diinput kedalam property.
- Aturan tata cara penamaan property sama dengan aturan penamaan variabel.

hp.php > ...

```php
<?php
class Hp {
    // isi dari class Hp.

    var $pemilik;
    var $merk;
    var $ukuran_layar;
    // isi dari class Hp.
}
?>
```

$pemilik, $merk, dan $ukuran_layar dan adalah property dari class Hp. penulisan property di dalam PHP sama dengan cara penulisan variabel, menggunakan tanda dollar ($). Sebuah **class** tidak harus memiliki property.

Method dalam Pemrograman Berbasis Objek

- **Method** adalah tindakan yang bisa dilakukan di dalam class. Jika menggunakan analogi class Hp kita, maka contoh method adalah: menghidupkan Hp, mematikan Hp dan berbagai tindakan lain.

- **Method** pada dasarnya adalah **function** yang berada di **dalam class**. Seluruh fungsi dan sifat function bisa diterapkan ke dalam method, seperti argumen/parameter, mengembalikan nilai (dengan keyword return), dan lain-lain.

methodhp.php > ...

```php
<?php
class Hp {
    function hidupkan_Hp() {
    //... isi dari method hidupkan_Hp
    }

    function matikan_Hp() {
    //... isi dari method matikan_Hp
    }

    //... isi dari class Hp
}
```

function **hidupkan_Hp()** dan function **matikan_Hp()** adalah method dari class Hp.
Penulisan method di dalam PHP sama dengan cara penulisan **function**.
Sebuah class tidak harus memiliki method.

Object dalam Pemrograman Berbasis Objek

- **Object** adalah hasil cetak dari class, atau hasil 'konkrit' dari class. Jika menggunakan analogi class Hp, maka objek dari class Hp bisa berupa: Hp_dia dan Hp_saya
- **Objek** dari class Hp akan memiliki seluruh ciri-ciri Hp, yaitu property dan method-nya.
- Proses **'mencetak'** objek dari class ini disebut dengan **'instansiasi'** *(instantiation).*
- Pada PHP, proses instansiasi dilakukan dengan menggunakan keyword **'new'.**
- **Hasil class** akan disimpan dalam variabel untuk selanjutnya digunakan dalam proses program.

objekhp.php > ...

```php
<?php
class Hp {
    //... isi dari class Hp
    }

$Hp_dia = new Hp();
$Hp_saya = new Hp();

var_dump($Hp_dia);
?>
```

$Hp_dia dan **$Hp_saya** merupakan objek dari **class Hp**.
Kedua objek ini akan memiliki seluruh property dan method dari **class Hp**.

Enkapsulasi *(encapsulation)*

- ***Encapsulation*** adalah sebuah metode untuk mengatur struktur class dengan cara menyembunyikan alur kerja dari class tersebut.

Enkapsulasi Objek:
Public, Protected dan Private

- Untuk membatasi hak akses kepada property dan method di dalam sebuah class, Kata kunci ini diletakkan sebelum nama property atau sebelum nama method.

Pengertian Hak Akses: **Public**

- Ketika sebuah property atau method dinyatakan sebagai public, maka seluruh kode program di luar class bisa mengaksesnya, termasuk class turunan.

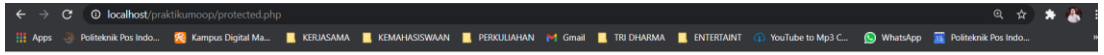
encapsulasi.php > ...

```php
<?php
// buat class Hp
class Hp {

    // buat public property
    public $pemilik;

    // buat public method
    public function hidupkan_Hp() {
        return "Hidupkan Hp";
    }
}

// buat objek dari class Hp (instansiasi)
$Hp_saya = new Hp();

// set property
$Hp_saya->pemilik="saya";

// tampilkan property
echo $Hp_saya->pemilik; // saya

// tampilkan method
echo $Hp_saya->hidupkan_Hp(); // "Hidupkan Hp"

```

Pengertian Hak Akses: **Protected**

- Jika sebuah property atau method dinyatakan sebagai **protected**, berarti property atau method tersebut **tidak bisa diakses dari luar class**, namun bisa diakses oleh **class itu sendiri** atau **turunan class** tersebut.

Dalam contoh di samping, pemanggilan property **$pemilik** dan method **hidupkan_Hp()** dari luar class akan menghasilkan error.



Fatal error: Uncaught Error: Cannot access protected property Hp::$pemilik in C:\xampp\htdocs\praktikumoop\protected.php:19 Stack trace: #0 {main} thrown in **C:\xampp\htdocs\praktikumoop\protected.php** on line **19**

protected.php > ...

```php
<?php

// buat class Hp
class Hp {

    // buat protected property
    protected $pemilik;

   // buat protected method
    protected function hidupkan_Hp() {
        return "Hidupkan Hp";
    }
}

// buat objek dari class Hp (instansiasi)
$Hp_saya = new Hp();

// set protected property akan menghasilkan error
$Hp_saya->pemilik="saya";
// Fatal error: Cannot access protected property Hp::$pemilik

// tampilkan protected property akan menghasilkan error
echo $Hp_saya->pemilik;
// Fatal error: Cannot access protected property Hp::$pemilik

// jalankan protected method akan menghasilkan error
echo $Hp_saya->hidupkan_Hp();
// Fatal error: Call to protected method Hp::hidupkan_Hp()
// from context
?>
```

Pengertian Hak Akses: **Protected**

- Walaupun akses level protected tidak bisa diakses dari luar class, namun bisa diakses dari dalam class itu sendiri,

property **$pemilik** di deklarasikan sebagai **protected**, sehingga pengaksesan dari luar class akan menghasilkan error.

Oleh sebeb itu buatlah sebuah **public method** yang akan menampilkan hasil property **$pemilik,** yakni method **akses_pemilik()**.

method **hidupkan_Hp()** yang tidak bisa diakses secara langsung. Saya menambahkan method **paksa_hidup()** yang secara internal akan mengakses method **hidupkan_Hp().**

protectedclass.php > ...

```php
<?php

// buat class Hp
class Hp {

    // buat protected property
    protected $pemilik="saya";

    public function akses_pemilik() {
        return $this->pemilik;
    }
    protected function hidupkan_Hp() {
        return "Hidupkan Hp";
    }
    public function paksa_hidup() {
        return $this->hidupkan_Hp();
    }
}

// buat objek dari class Hp (instansiasi)
$Hp_saya = new Hp();

// jalankan method akses_pemilik()
echo $Hp_saya->akses_pemilik(); // "saya"

// jalankan method paksa_hidup()
echo $Hp_saya->paksa_hidup(); // "Hidupkan Hp"
?>
```

Selain dari dalam **class itu sendiri**, **property** dan **method** dengan hak akses **protected** juga bisa diakses dari **class turunan.**

protectturunan.php > ...

```php
<?php

// buat class smaptphone
class smaptphone{

    // property dengan hak akses protected
    protected $kualitas_kamera = "Kameranya bening seperti kamu, hehe..";
}

// buat class Hp
class Hp extends smaptphone{
    public function tampilkan_kamera() {
        return $this->kualitas_kamera;
    }
}

// buat objek dari class Hp (instansiasi)
$Hp_baru = new Hp();

// jalankan method
echo $Hp_baru->tampilkan_kamera(); // "Kameranya bening seperti kamu, hehe.."
?>
```

Pada kode di samping, walaupun method **$kualitas_kamera** di set sebagai **protected** pada ***class smartphone***, tetapi masih bisa diakses dari ***class Hp*** yang merupakan turunan dari ***class smartphone***.

Menggunakan istilah ***extends***

Pengertian Hak Akses: **Private**

- Hak akses terakhir dalam konsep enkapsulasi adalah **private.** Jika sebuah **property** atau **method di-set** sebagai **private**, maka satu-satunya **yang bisa mengakses** adalah **class itu sendiri**.
- Class lain tidak bisa mengaksesnya, termasuk class turunan.

protectedprivate.php > ...

```php
<?php

// buat class smartphone
class smartphone {

    // property dengan hak akses protected
    private $kualitas_kamera = "Kameranya bening seperti kamu, hehe..";

    public function tampilkan_kamera() {
        return $this->kualitas_kamera;
    }
}

// buat class Hp
class Hp extends smartphone{

    public function tampilkan_kamera() {
        return $this->kualitas_kamera;
    }
}

// buat objek dari class Hp (instansiasi)
$smartphone_baru = new smartphone();
$Hp_baru = new Hp();

// jalankan method dari class smartphone
echo $smartphone_baru->tampilkan_kamera(); // "Kameranya bening seperti kamu, hehe.."

// jalankan method dari class Hp (error)
echo $Hp_baru->tampilkan_kamera();
// Notice: Undefined property: Hp::$kualitas_kamera
?>
```

- **Class Hp** merupakan **turunan dari class smartphone**.
- Di dalam class smartphone terdapat property **$kualitas_kamera** dengan akses **level private**. Di dalam **class smartphone** dan **class Hp**.
- Pengaksesan method **tampilkan_kamera()** dari objek **$Hp_baru** sukses ditampilkan karena berada di dalam satu class dimana property **$kualitas_kamera** berada.
- Jika method **tampilkan_kamera()** diakses dari objek **$Hp_baru** yang merupakan turunan dari class smartphone, PHP akan mengeluarkan error karena property $kualitas_kamera tidak dikenal.
- Akses level private sering digunakan untuk menyembunyikan property dan method agar tidak bisa diakses di luar class.

protectedprivate.php > ...

```php
<?php

// buat class smartphone
class smartphone {

    // property dengan hak akses protected
    private $kualitas_kamera = "Kameranya bening seperti kamu, hehe..";

    public function tampilkan_kamera() {
        return $this->kualitas_kamera;
    }
}

// buat class Hp
class Hp extends smartphone{

    public function tampilkan_kamera() {
        return $this->kualitas_kamera;
    }
}

// buat objek dari class Hp (instansiasi)
$smartphone_baru = new smartphone();
$Hp_baru = new Hp();

// jalankan method dari class smartphone
echo $smartphone_baru->tampilkan_kamera(); // "Kameranya bening seperti kamu, hehe.."

// jalankan method dari class Hp (error)
echo $Hp_baru->tampilkan_kamera();
// Notice: Undefined property: Hp::$kualitas_kamera
?>
```

Any **Question?**

Roni Andarsyah, ST., M.Kom
Lecture Series