PHP Object-Oriented Programming (OOP)

OOP adalah singkatan dari Object-Oriented Programming. OOP adalah teknik pemrograman yang berorientasi pada objek, objek tersebut akan berisi data dan fungsi. OOP memiliki beberapa keunggulan dibandingkan pemrograman prosedural, yaitu sebagai berikut:

- **OOP lebih cepat dan mudah dijalankan.**
- **OOP memberikan struktur yang jelas.**
- **OOP membuat kode lebih mudah di-maintain, dimodifikasi dan di-debug.**
- **OOP memungkinkan waktu pengembangan web lebih singkat.**

Kelas dan objek adalah dua aspek utama dari OOP.

1. Kelas dan Objek

Kelas merupakan template (cetakan) dari objek dan objek adalah instance (bentuk konkret) dari kelas. Ketika objek diinstansiasi/dibuat, maka akan mewarisi semua atribut dan method dari kelas, tetapi setiap objek akan memiliki nilai yang berbeda untuk atribut tersebut.

Kelas didefinisikan dengan menggunakan keyword class, diikuti dengan nama kelas dan tanda kurung kurawal ({}). Sintaks untuk mendefinisikan kelas adalah sebagai berikut:

```php
<?php
class <nama-kelas> {
    //Serangkaian kode program
}
?>
```

Contohnya terdapat kelas mahasiswa yang memiliki atribut nama, dan umur. Selain itu terdapat juga method untuk memberi nilai atribut dan mengambil data atribut.

```php
<?php
class Mahasiswa {
    public $nama;
    public $umur;

    public function setNama($nama){
        $this->nama = $nama;
    }
    public function getNama(){
        return $this->nama;
    }
    public function setUmur($umur){
        $this->umur = $umur;
    }
    public function getUmur(){
        return $this->umur;
    }
}
?>
```

Sebelumnya sudah dijelaskan bahwa kelas merupakan template. Oleh karena itu agar bisa digunakan harus dibuat objek dari kelas tersebut.
Untuk membuat objek menggunakan keyword new. Berikut contoh pembuatan objek dari kelas Mahasiswa:

```php
<?php
class Mahasiswa {
    public $nama;
    public $umur;

    public function setNama($nama){
        $this->nama = $nama;
    }
    public function getNama(){
        return $this->nama;
    }
    public function setUmur($umur){
        $this->umur = $umur;
    }
    public function getUmur(){
        return $this->umur;
    }
}
$mhs1 = new Mahasiswa();
$mhs1->setNama('Udin');
$mhs1->setUmur(22);

echo $mhs1->getNama();
echo "<br>";
echo $mhs1->getUmur();

?>
```

2. Konstruktor

Konstruktor memungkinkan untuk menginisiasi atribut objek saat pembuatan objek. Untuk menggunakan konstruktor yaitu dengan *fungsi __construct()*. Fungsi tersebut otomatis dipanggil saat pembuatan objek. Untuk contoh, modifikasi kelas mahasiswa dengan menambahkan konstruktor.

```php
<?php
class Mahasiswa {
    public $nama;
    public $umur;

    public function __construct($nama, $umur){
        $this->nama = $nama;
        $this->umur = $umur;
    }
      // public function setNama($nama){
      //   $this->nama = $nama;
```

```
    // }

    public function getNama(){
        return $this->nama;
    }
    // public function setUmur($umur){
    //     $this->umur = $umur;
    // }
    public function getUmur(){
        return $this->umur;
    }
}
$mhs1 = new Mahasiswa("Sandi", 22);
echo $mhs1->getNama();
echo "<br>";
echo $mhs1->getUmur();
```

Perhatikan kode program diatas, dengan konstruktor dapat mengisi nilai atribut saat pembuatan objek.

3. Access Modifiers

Atribut dan method dapat memiliki access modifiers yang mengontrol pengaksesan. Terdapat tiga access modifiers, yaitu:

- ***Public*-atribut dan method dapat diakses dari mana saja. Public merupakan *access modifiers default*.**
- ***Protected*-atribut dan method dapat diakses di dalam kelas dan kelas yang diturunkan dari kelas itu.**
- ***Private*-atribut dan method dapat diakses di dalam kelas itu saja.**

Untuk contoh terdapat 3 atribut dengan access modifiers yang berbeda.

```
<?php
class Penduduk {
    public $namaP;
    protected $umurP;
    private $alamatP;
}
$pddk = new Penduduk();
echo $pddk->namaP = "Andri"; //Berjalan
echo $ppdk->umurP = 22; // Error
echo $pddk->alamatP = "Jakarta Pusat"; //Error
?>
```

Perhatikan kode program diatas, jika menggunakan *access modifiers protected* dan *private* akan *error* saat diakses langsung. Oleh karena itu jika ingin mengakses atribut dengan access modifiers protected atau private, perlu (*method setter,* untuk mengisi nilai atribut*)* dan *(method getter* untuk mengambil nilai dari atribut*).*

```php
<?php
class Penduduk {
    public $namaP;
    protected $umurP;
    private $alamatP;

    public function setUmurP($umurP){
        $this->umurP = $umurP;
    }
    public function getUmurP(){
        return $this->umurP;
    }
    public function setAlamatP($alamatP){
        $this->alamatP = $alamatP;
    }
    public function getAlamatP(){
        return $this->alamatP;
    }
}
$pddk = new Penduduk();
$pddk->setUmurP(43);
$pddk->setAlamatP("Jakarta Barat");
echo $pddk->namaP = "Andri"; //Berjalan
echo $pddk->getUmurP(); // Berjalan
echo $pddk->getAlamatP(); // Berjalan
?>
```

4. Inheritance

Inheritance merupakan konsep pewarisan, dimana sebuah kelas dapat menurunkan atribut dan method kepada kelas lain. Hanya atribut dan method dengan access modifiers public serta protected yang diturunkan. Kelas hasil inheritance dapat memiliki atribut dan method sendiri. Penggunaan inheritance menggunakan keyword extends.

Sebagai contoh adalah kelas Penduduk dengan atribut nama, umur, dan alamat. Selain itu ada kelas Bansos yang akan mewarisi atribut kelas Penduduk (nama dan umur) dengan tambahan atribut yaitu status.

```php
<?php
class Penduduk {
    public $nama;
    protected $umur;
    private $alamat = "Jakarta";
}

class Bansos extends Penduduk {
    public $nama = "Anton";
    protected $umur = 24;
    public $status = "Miskin";
    // private $alamat = "Jakarta";
    public function showPenduduk(){
        echo $this->nama."<br>";
        echo $this->status."<br>";
        echo $this->umur."<br>";
        echo $this->alamat."<br><br>";
    }
}
$penduduk = new Bansos();
$penduduk->showPenduduk(); // Menampilkan Anton, Miskin, 24, Undefined
property
echo $penduduk->nama."<br>"; // Anton
echo $penduduk->status."<br>"; // Miskin
echo $penduduk->umur."<br>"; // Fatal Error
echo $penduduk->alamat."<br>" // Undefined
?>
```

Perhatikan kode program diatas. Ketika kelas Bansos mewarisi kelas Penduduk, kita dapat mengakses atribut kelas dengan access modifiers public dan protected yaitu atribut nama dan umur dari kelas Penduduk. Namun ketika kita membuat objek dari kelas Bansos, atribut nama dapat diakses secara langsung sedangkan atribut umur tidak bisa diakses secara langsung (echo $penduduk->umur akan menghasilkan Fatal error). Untuk mengakses atribut umur harus dibuat method dengan access modifiers public (public function showPenduduk()). Namun atribut private $alamat pada kelas Penduduk tidak dapat diakses oleh kelas Bansos maupun dengan access modifiers public ($penduduk->showPenduduk() menghasilkan undefined pada bagian alamat, echo $penduduk->alamat juga menghasilkan undefined).