Syukrillah - 22552011247

Konsep OOP (Object-Oriented Programming)

Sumber: **DESIGN SYSTEM - 2023 EDITION (ByteByteGo) - 8 Key OOP Concepts Every Developer Should Know (halaman 257 - 258)**

1. Kelas (Class):

- **Kelas adalah cetak biru atau templat untuk membuat objek.** Ini mendefinisikan atribut (data) dan metode (fungsi) yang akan dimiliki oleh objek-objek yang dibuat dari kelas tersebut.
- Bayangkan kelas sebagai resep kue. Resepnya (kelas) menjelaskan bahan-bahan (atribut) dan langkah-langkah (metode) untuk membuat kue. Setiap kue yang Anda buat dari resep tersebut adalah objek.

2. Objek (Object):

- **Objek adalah instansiasi dari kelas.** Ini adalah entitas nyata yang memiliki atribut dan perilaku yang ditentukan oleh kelasnya.
- Dalam contoh resep kue, setiap kue yang Anda buat adalah objek. Setiap objek memiliki karakteristik unik (misalnya, rasa, ukuran) tetapi mengikuti struktur yang ditentukan oleh kelasnya (resep).

3. Enkapsulasi (Encapsulation):

- **Enkapsulasi adalah konsep pembungkusan data dan metode yang bekerja pada data tersebut dalam satu unit, yaitu objek.**
- **Ini menyembunyikan detail internal objek dari dunia luar dan hanya menyediakan antarmuka publik untuk berinteraksi dengan objek.**
- **Tujuan enkapsulasi adalah untuk melindungi data dari akses yang tidak sah dan untuk menjaga integritas data.**
- Sebagai contoh, bayangkan sebuah mesin kopi. Anda hanya perlu menekan tombol untuk membuat kopi, tanpa perlu mengetahui detail internal bagaimana mesin tersebut bekerja. Detail internal mesin kopi tersebut di enkapsulasi.

4. Pewarisan (Inheritance):

- **Pewarisan memungkinkan suatu kelas (kelas anak) untuk mewarisi atribut dan metode dari kelas lain (kelas induk).**
- **Ini memungkinkan pembuatan hierarki kelas dan penggunaan kembali kode.**
- Contohnya, kelas "Kendaraan" dapat menjadi kelas induk, dan kelas "Mobil" dan "Motor" dapat menjadi kelas anak yang mewarisi atribut dan metode dari kelas "Kendaraan".

5. Polimorfisme (Polymorphism):

- **Polimorfisme memungkinkan objek dari kelas yang berbeda untuk merespons metode yang sama dengan cara yang berbeda.**
- **Ini memberikan fleksibilitas dalam pengembangan kode dan memungkinkan pembuatan kode yang lebih umum.**
- Contohnya, metode "bersuara" dapat memiliki implementasi yang berbeda di kelas "Anjing" (menggonggong) dan kelas "Kucing" (mengeong).

6. Asosiasi (Association):

- Asosiasi adalah hubungan antara dua kelas yang menunjukkan bahwa objek dari satu kelas menggunakan objek dari kelas lain.
- Hubungan ini bisa berupa "menggunakan", "memiliki", atau "mengetahui".

7. Agregasi (Aggregation):

- **Agregasi adalah jenis asosiasi khusus yang menunjukkan hubungan "memiliki-a" antara dua kelas.**
- **Dalam agregasi, objek dari satu kelas adalah bagian dari objek dari kelas lain, tetapi objek-objek tersebut dapat eksis secara independen.**

- Contohnya, kelas "Jurusan" memiliki agregasi dengan kelas "Mahasiswa", karena jurusan "memiliki" mahasiswa, tetapi mahasiswa dapat eksis tanpa jurusan.

8. Komposisi (Composition):

- Komposisi adalah jenis agregasi khusus yang menunjukkan hubungan "memiliki-a" yang kuat antara dua kelas.
- Dalam komposisi, objek dari satu kelas adalah bagian integral dari objek dari kelas lain dan tidak dapat eksis secara independen.
- Contohnya, kelas "Mobil" memiliki komposisi dengan kelas "Mesin", karena mesin adalah bagian integral dari mobil dan tidak dapat eksis tanpa mobil.

Pentingnya OOP:

- **OOP membantu dalam mengorganisasi dan mengelola kode yang kompleks.**
- **OOP meningkatkan penggunaan kembali kode, yang menghemat waktu dan upaya pengembangan.**
- **OOP membuat kode lebih mudah dipelihara dan diubah.**
- **OOP memodelkan dunia nyata dengan lebih baik, yang membuat kode lebih intuitif dan mudah dipahami.**

Contoh OOP di Pemograman Web

Studi Kasus: Sistem Reservasi Hotel

---

1. Class (Kelas)

```
class Kamar {
    public $nomor;
    public $tipe;
    public $hargaPerMalam;

    public function infoKamar() {
        return "Kamar $this->nomor ($this->tipe) - Rp $this-
>hargaPerMalam/malam";
    }
}
```

---

2. Object (Objek)

```php
$kamar1 = new Kamar();
$kamar1->nomor = 101;
$kamar1->tipe = "Deluxe";
$kamar1->hargaPerMalam = 500000;

echo $kamar1->infoKamar(); // Output: Kamar 101 (Deluxe) - Rp 500000/malam
```

---

3. Encapsulation (Enkapsulasi)

```php
class Tamu {
    private $nama;

    public function setNama($nama) {
        $this->nama = $nama;
    }

    public function getNama() {
        return $this->nama;
    }
}

$tamu = new Tamu();
$tamu->setNama("Budi");
echo $tamu->getNama(); // Output: Budi
```

---

4. Inheritance (Pewarisan)

```php
class User {
    protected $username;

    public function login() {
        return "$this->username berhasil login.";
    }
}

class Admin extends User {
    public function setUsername($username) {
        $this->username = $username;
    }
}
```

```php
$admin = new Admin();
$admin->setUsername("admin_hotel");
echo $admin->login(); // Output: admin_hotel berhasil login.
```

---

5. Polymorphism (Polimorfisme)

```php
class Pembayaran {
    public function proses() {
        return "Memproses pembayaran...";
    }
}

class PembayaranKartu extends Pembayaran {
    public function proses() {
        return "Pembayaran dengan kartu kredit diproses.";
    }
}

class PembayaranTransfer extends Pembayaran {
    public function proses() {
        return "Pembayaran melalui transfer bank diproses.";
    }
}

function bayar(Pembayaran $pembayaran) {
    echo $pembayaran->proses();
}

bayar(new PembayaranKartu());     // Output: Pembayaran dengan kartu kredit
diproses.
bayar(new PembayaranTransfer());  // Output: Pembayaran melalui transfer
bank diproses.
```

---

6. Association (Asosiasi)

```php
class Reservasi {
    public $tamu;

    public function __construct(Tamu $tamu) {
        $this->tamu = $tamu;
    }

    public function detail() {
        return "Reservasi atas nama: " . $this->tamu->getNama();
```

```
    }
}

$reservasi = new Reservasi($tamu);
echo $reservasi->detail(); // Output: Reservasi atas nama: Budi
```

---

7. Aggregation (Agregasi)

```
class Hotel {
    public $nama;
    public $kamar = []; // daftar kamar (bisa hidup tanpa hotel)

    public function tambahKamar(Kamar $kamar) {
        $this->kamar[] = $kamar;
    }
}

$hotel = new Hotel();
$hotel->nama = "Hotel Mewah";
$hotel->tambahKamar($kamar1);
```

---

8. Composition (Komposisi)

```
class Mesin {
    public function nyalakan() {
        return "Mesin dinyalakan.";
    }
}

class Mobil {
    private $mesin;

    public function __construct() {
        $this->mesin = new Mesin(); // Mesin hanya hidup jika Mobil dibuat
    }

    public function jalan() {
        return $this->mesin->nyalakan() . " Mobil berjalan.";
    }
}

$mobil = new Mobil();
echo $mobil->jalan(); // Output: Mesin dinyalakan. Mobil berjalan.
```