Object-Oriented Programming

Object-Oriented Programming (OOP)

- Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang berfokus pada objek dan data daripada fungsi dan logika.
- OOP berkaitan dengan pemodelan sebuah sistem sebagai kumpulan objek, di mana setiap objek mewakili aspek tertentu dari sistem.

Keuntungan OOP

- Memungkinkan pemisahan concerns dan abstraksi.
- Meningkatkan modularitas dan pemeliharaan (maintenance) kode.
- Objek dapat digunakan ulang dan diwariskan.
- Membantu dalam memodelkan sistem sesuai dengan keadaan nyata.

Konsep Utama

1. **Objek**: Representasi dari entitas dalam sistem.
2. **Metode**: Fungsi atau perilaku yang dimiliki oleh objek.
3. **Data**: Informasi atau atribut yang dimiliki oleh objek.
4. **Enkapsulasi**: Menyembunyikan detail internal objek.
5. **Pewarisan**: Mewariskan properti dan metode dari class lain.
6. **Polimorfisme**: Kemampuan objek untuk memiliki banyak bentuk.

Studi Kasus

Ketika kita memodelkan suatu masalah dalam bentuk objek dalam OOP, kita membuat definisi abstrak yang mewakili jenis objek yang ingin kita miliki dalam sistem kita.

Misalnya, jika kita sedang memodelkan sebuah sekolah, kita mungkin ingin memiliki objek yang mewakili profesor. Setiap profesor memiliki beberapa properti yang sama: mereka semua memiliki nama dan subjek yang mereka ajar.

Selain itu, setiap profesor dapat melakukan hal tertentu: mereka semua dapat menilai tugas dan mereka dapat memperkenalkan diri mereka kepada siswa mereka di awal tahun.

Studi Kasus

```
class Professor
    properties
        name
        teaches
    methods
        grade(assignment)
        introduceSelf()
```

Dalam contoh ini, kita membuat definisi abstrak dari apa yang kita inginkan dari objek profesor. Kita tidak mempedulikan detail implementasi, kita hanya ingin tahu apa yang dapat dilakukan oleh objek profesor.

- Memiliki properti `name` dan `teaches`.
- Memiliki metode `grade()` untuk menilai tugas dan `introduceSelf()` untuk memperkenalkan diri.

Studi Kasus

- Class `Professor` adalah definisi abstrak dari apa yang kita inginkan dari objek profesor.
- Jika berdiri sendiri, class ini tidak melakukan apa-apa.
- Class ini adalah sebuah template atau blueprint yang dapat digunakan untuk membuat objek profesor.
- Class ini disebut class induk, superclass, atau abstract class.

Studi Kasus

- Untuk membuat objek baru, kita memerlukan fungsi khusus yaitu `constructor`.
- Setiap objek yang dibuat dari abstract class disebut sebagai **instance** atau subclass.
- Proses pembentukan objek dari class disebut **instansiation**.
- Objek yang dibuat dari class `Professor` disebut sebagai instance dari class `Professor`.

Studi Kasus

```
class Professor
  properties
    name
    teaches
  constructor
    Professor(name, teaches)
  methods
    grade(assignment)
    introduceSelf()
```

Studi Kasus

```
john = new Professor('Walsh', 'Psychology')
jane = new Professor('Lillian', 'Poetry')

john.teaches // 'Psychology'
john.introduceSelf() // 'My name is Professor Walsh.'

jane.teaches // 'Poetry'
jane.introduceSelf() // 'My name is Professor Lillian.'
```

Objek dalam JavaScript

- Dalam JavaScript, objek adalah koleksi pasangan kunci-nilai (key-value).
- Objek dapat mewakili entitas nyata dan memiliki properti dan metode.

```
const person = {
  name: 'John',
  age: 30,

  greet() {
    console.log(`Hello, my name is ${this.name}.`)
  },
}
```

Class dalam JavaScript

- Dalam ES6 (ECMAScript 2015) dan seterusnya, Anda dapat menggunakan sintaks `class` untuk mendefinisikan class.
- Class berperan sebagai blueprint untuk membuat objek.

Class dalam JavaScript

```
class Person {
  constructor(name, age) {
    this.name = name
    this.age = age
  }

  greet() {
    console.log(`Hello, my name is ${this.name}.`)
  }
}

const person = new Person('John', 30)
person.greet()
```

Constructor dan Instance

- Method `constructor` dipanggil saat objek baru dibuat dari class.
- Keyword `new` digunakan untuk membuat objek baru dari class.
- Instansiasi adalah proses membuat objek dari class.
- Instance adalah objek yang dibuat dari class.

Constructor dan Instance

```
class Circle {
  constructor(radius) {
    this.radius = radius
  }

  area() {
    return Math.PI * this.radius * this.radius
  }
}

const circle1 = new Circle(5)
const circle2 = new Circle(8)

console.log(circle1.area()) // 78.54
console.log(circle2.area()) // 201.06
```

Inheritance

- Inheritance adalah proses di mana sebuah class mewarisi properti dan metode dari class lain.
- JavaScript mendukung pewarisan tunggal menggunakan kata kunci `extends`.

Inheritance

```
class Animal {
  constructor(name) {
    this.name = name
  }

  speak() {
    console.log(`${this.name} makes a sound.`)
  }
}
```

```
class Bird extends Animal {}

const tweety = new Bird('Tweety')
tweety.speak() // Tweety makes a sound.
```

Method Overriding

- Penggantian metode adalah konsep dalam pemrograman berorientasi objek di mana sebuah class anak (subclass) dapat menggantikan (override) metode yang telah didefinisikan dalam class induk (superclass).
- Saat metode di class anak memiliki nama, parameter, dan tipe kembalian yang sama seperti metode di class induk, kita menggantikan perilaku metode tersebut.

Method Overriding

```
class Dog extends Animal {
  speak() {
    console.log(`${this.name} barks.`)
  }
}

const dog = new Dog('Buddy')
dog.speak() // Buddy barks.
```

Latihan

- Definisikan class `BankAccount` dengan properti seperti `accountNumber`, `accountHolder`, dan `balance`.
- Implementasikan method di class `BankAccount` untuk melakukan deposit, penarikan, dan pengecekan saldo.
- Buat class `Bank` yang dapat menyimpan dan mengelola koleksi akun bank.
- Implementasikan method di class `Bank` untuk menambahkan akun, menghapus akun, dan menampilkan semua akun.