



# **MATERI AJAR**

**Pemrograman Berorientasi Objek**

**Materi Class dan Object**



## **1. Pengenalan Dasar Pemrograman Berorientasi Objek**

### **a. Paradigma Pemrograman Berorientasi Objek**

Pemrograman Berorientasi Objek (Object-Oriented Programming atau OOP) adalah paradigma pemrograman yang menggunakan "objek" sebagai elemen dasar dalam penyusunan program. Objek-objek ini berisi data (atribut) dan perilaku (metode) yang merepresentasikan entitas nyata di dunia. Menurut (Siregar & Harahap, 2023), OOP adalah pendekatan pemrograman yang berorientasi pada objek, di mana semua data dan fungsi dibungkus dalam kelas-kelas atau objek-objek. Setiap objek dapat menerima pesan, memproses data, mengirim, menyimpan, dan memanipulasi data.

Secara umum, OOP mencakup konsep-konsep utama seperti enkapsulasi, pewarisan, dan polimorfisme. Enkapsulasi adalah mekanisme pemrograman yang membungkus data dan fungsi dalam satu unit, yaitu kelas, untuk menyembunyikan detail implementasi dan hanya menampilkan antarmuka yang diperlukan. Pewarisan memungkinkan sebuah kelas mewarisi atribut dan metode dari kelas lain, sehingga mendukung hierarki dan pengorganisasian kode yang lebih baik. Polimorfisme memungkinkan objek untuk mengambil banyak bentuk, sehingga memudahkan programmer dalam membuat referensi pemanggilan objek dari satu metode.

Objek adalah suatu entitas yang memiliki atribut dan karakteristik tertentu. Sementara itu, paradigma Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) merupakan pendekatan dalam pengembangan perangkat lunak yang menitikberatkan pada pemecahan program menjadi objek-objek yang dapat saling berinteraksi. Menurut Wegner (1990), pemrograman prosedural dirancang sebagai serangkaian langkah aksi yang harus dijalankan secara berurutan. Sebaliknya, dalam pemrograman berorientasi objek, program terdiri dari berbagai komponen yang saling berhubungan, di mana setiap komponen memiliki peran spesifik yang dapat diakses melalui antarmukanya. Pendekatan ini meniru hubungan dan interaksi antar objek di dunia nyata dalam suatu konteks aplikasi tertentu.

### **b. Perbedaan Pemrograman Berorientasi Objek dengan Pemrograman Prosedural**

Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) dan Pemrograman Prosedural merupakan dua paradigma pemrograman yang

memiliki pendekatan berbeda dalam pengorganisasian dan pengelolaan kode program.

Aspek	Pemrograman Prosedural	Pemrograman Berorientasi Objek (OOP)
<b>Pendekatan</b>	Berbasis langkah-langkah atau prosedur (fungsi).	Berbasis objek yang berisi data dan perilaku.
<b>Struktur Program</b>	Program dibagi menjadi fungsi dan prosedur.	Program dibagi menjadi class dan objek.
<b>Data dan Aksesibilitas</b>	Data bersifat global dan dapat diakses oleh semua bagian program.	Data dikemas dalam objek dengan mekanisme enkapsulasi.
<b>Keamanan Data</b>	Rentan terhadap perubahan tidak disengaja karena data bersifat global.	Lebih aman karena data dapat dikendalikan dengan access modifier (private, public, protected).
<b>Reusability (Penggunaan Ulang)</b>	Fungsi harus ditulis ulang jika ingin digunakan kembali.	Mendukung pewarisan (inheritance) sehingga kode lebih reusable.
<b>Fleksibilitas dan Skalabilitas</b>	Kurang fleksibel dan sulit dikembangkan untuk proyek besar.	Lebih fleksibel, modular, dan cocok untuk proyek besar.
<b>Contoh Bahasa</b>	C, Pascal, COBOL.	Java, C#, Python, C++.

## 2. Class dan Object

### a. Class

Dalam Pemrograman Berorientasi Objek (OOP), class adalah cetak biru atau blueprint yang digunakan untuk membuat objek. Class mendefinisikan atribut (data) dan metode (fungsi/perilaku) yang akan dimiliki oleh objek. Dengan kata lain, class berfungsi sebagai template yang mendeskripsikan bagaimana suatu objek akan dibuat dan berperilaku.

Bayangkan class sebagai cetak biru rumah dan objek sebagai rumah yang dibangun berdasarkan cetak biru tersebut. Dari satu cetak biru, kita bisa membuat banyak rumah dengan warna dan ukuran berbeda, tetapi memiliki struktur yang sama.

Berikut contoh bentuk umum dari class:

```
// Deklarasi class
[access_modifier] class NamaClass
{
    // Atribut (fields)
    [access_modifier] TipeData namaVariabel;

    // Konstruktor (opsional)
    public NamaClass()
    {
        // Inisialisasi atribut jika diperlukan
    }

    // Metode (fungsi) dalam class
    [access_modifier] TipeKembalian NamaMetode([parameter])
    {
        // Isi metode
    }
}
```

1. Nama Class tersebut adalah **NamaClass**
2. access\_modifier → Menentukan tingkat aksesibilitas class, atribut, atau metode. Beberapa modifier yang umum digunakan:
  - public → Dapat diakses dari mana saja.
  - private → Hanya dapat diakses di dalam class itu sendiri.
  - protected → Dapat diakses oleh class itu sendiri dan turunannya (inheritance).
  - internal → Dapat diakses dalam satu assembly/proyek yang sama.
3. Atribut (Fields/Variabel) → Variabel yang menyimpan data untuk objek.

Contoh

4. Konstruktor → Metode khusus yang dipanggil saat objek dibuat.
5. Metode → Berisi perilaku atau aksi yang dapat dilakukan oleh objek.

#### b. Object

Di dalam C#, sebuah class dideklarasikan menggunakan kata kunci class. Berikut adalah struktur dasar sebuah class dalam bahasa C#:

```
// Mendeklarasikan class bernama Mobil
public class Mobil
{
    // Atribut (data)
    public string merk;
    public string warna;

    // Metode (fungsi/perilaku)
    public void Jalan()
    {
        Console.WriteLine("Mobil sedang berjalan...");
    }

    public void Berhenti()
    {
        Console.WriteLine("Mobil telah berhenti.");
    }
}
```

Pada contoh di atas:

- Mobil adalah class yang mendefinisikan dua atribut (merk dan warna).
- Jalan() dan Berhenti() adalah metode yang menentukan perilaku dari class Mobil.

Setelah kita mendefinisikan sebuah class, kita dapat membuat objek dari class tersebut menggunakan kata kunci new. Berikut adalah contoh penggunaan class Mobil:

```
using System;

class Program
{
    static void Main()
    {
        // Membuat objek dari class Mobil
        Mobil mobilA = new Mobil();

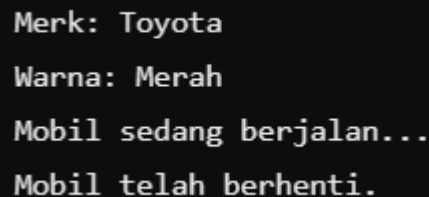
        // Memberikan nilai atribut
        mobilA.merk = "Toyota";
        mobilA.warna = "Merah";

        // Menampilkan informasi objek
        Console.WriteLine("Merk: " + mobilA.merk);
        Console.WriteLine("Warna: " + mobilA.warna);

        // Memanggil metode dari objek
        mobilA.Jalan();
        mobilA.Berhenti();
    }
}
```

1. Membuat objek: `Mobil mobilA = new Mobil();` → membuat objek mobilA dari class Mobil.
2. Mengisi atribut: `mobilA.merk = "Toyota";` dan `mobilA.warna = "Merah";` → memberikan nilai pada atribut.
3. Menampilkan atribut: `Console.WriteLine()` digunakan untuk mencetak nilai atribut.
4. Memanggil metode: `mobilA.Jalan();` dan `mobilA.Berhenti();` → memanggil fungsi yang ada di dalam class Mobil.

Jika program dijalankan, hasilnya akan seperti berikut:



```
Merk: Toyota
Warna: Merah
Mobil sedang berjalan...
Mobil telah berhenti.
```

## Kesimpulan

1. Class adalah cetak biru (blueprint) untuk membuat objek. Class mendefinisikan atribut (data) dan metode (perilaku) yang akan dimiliki oleh objek.
2. Object adalah instance dari sebuah class. Objek dibuat berdasarkan class dan dapat memiliki nilai atribut yang berbeda tetapi tetap menggunakan metode yang sama.
3. Setiap objek yang dibuat dari class yang sama bisa memiliki karakteristik (atribut) yang berbeda, tetapi tetap mengikuti aturan yang telah didefinisikan dalam class tersebut.
4. Membuat objek dalam C# menggunakan kata kunci `new` dan objek tersebut dapat mengakses atribut serta metode yang telah didefinisikan dalam class.
5. Konsep class dan object dalam OOP memudahkan pengelolaan program dengan cara membagi tugas program ke dalam unit-unit yang lebih kecil dan dapat digunakan kembali, sehingga kode lebih terstruktur, modular, dan mudah dikembangkan.



# LKPD

**Pemrograman Berorientasi Objek**

**Materi Class dan Object**



Mata Pelajaran	Dasar-dasar Pemrograman
Materi	Class, Object, dan Namespace
Kelas/Semester	X/2
Alokasi Waktu	45 Menit
Hari/Tanggal	
Nama	
Kelas	

### Soal A

1. Membuat kelas Siswa

Class **Siswa** akan memiliki tiga atribut: **nama**, **kelas**, dan **nilaiRataRata**. Selain itu, class ini juga akan memiliki metode untuk menampilkan informasi tentang siswa.

2. Setelah class Siswa didefinisikan, kita akan membuat objek dari class tersebut di dalam class Main. Setiap objek akan mewakili satu Siswa.



**Soal B**

1. Jelaskan dengan bahasa sendiri apa yang dimaksud dengan class dan objek dalam pemrograman berorientasi objek!
2. Mengapa metode penting dalam sebuah class? Berikan contoh sederhana!
3. Buatlah sebuah class dan objek untuk siswa!
4. Bagaimana hubungan antara class dan objek dalam membangun suatu program? Jelaskan dengan contoh!
5. Apa perbedaan antara atribut dan metode dalam sebuah class? Berikan penjelasan dan contoh!