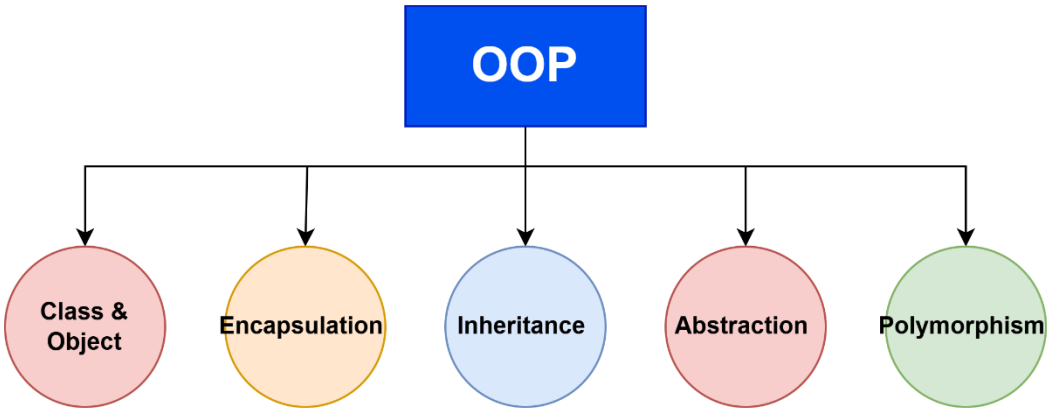
MODUL 4

OBJECT ORIENTED PROGRAMMING JAVA I

4.1 Pengenalan OOP

OOP (Object-Oriented Programming) atau Pemrograman Berorientasi Objek adalah paradigma pemrograman yang berfokus pada penggunaan objek untuk merancang dan mengimplementasikan program. OOP membantu dalam membuat kode yang lebih terstruktur, mudah dikelola, dan dapat digunakan kembali.



4.2 Konsep Dasar OOP

a) Class

- Dalam Object-Oriented Programming (OOP), kelas (class) adalah kerangka kerja (blueprint) yang digunakan untuk menciptakan objek. Kelas berfungsi seperti cetakan yang mendefinisikan atribut (data/properti) dan metode (fungsi/perilaku) dari objek yang dibuat berdasarkan kelas tersebut.
- Contoh: kelas mahasiswa, kelas kendaraan, kelas hewan, dll.

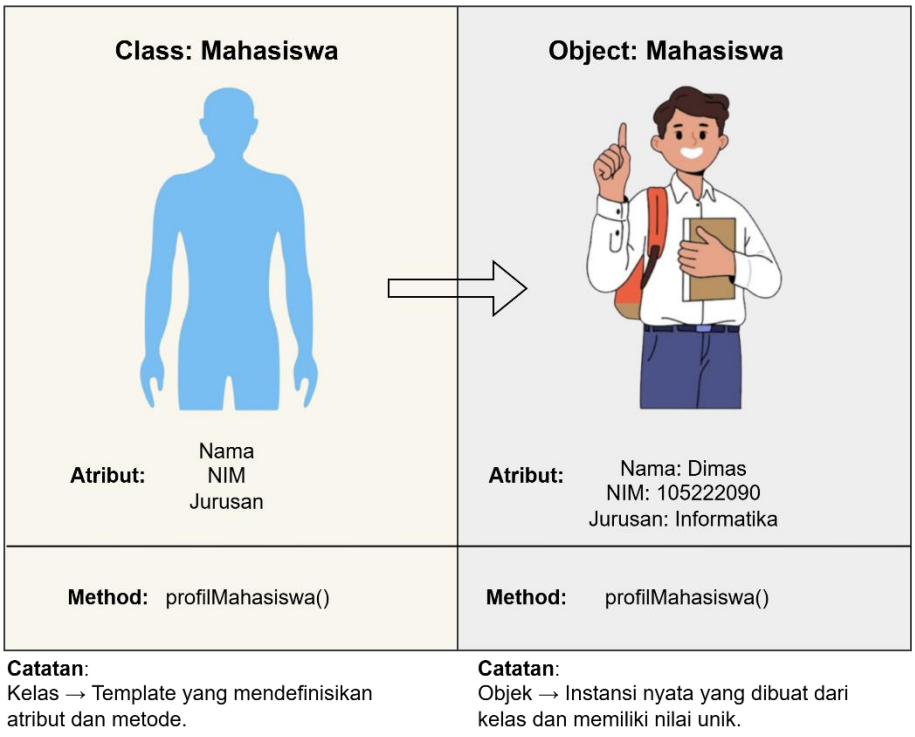
b) Object

Dalam pemrograman berorientasi objek (OOP), objek adalah entitas yang memiliki dua karakteristik utama:

- State (Keadaan/Status) → Merujuk pada atribut atau properti yang dimiliki oleh objek.
- Behavior (Perilaku) → Merujuk pada aksi atau metode yang dapat dilakukan oleh objek.

Contoh objek dalam dunia nyata:

- Buku memiliki state (judul, penulis, jumlah halaman) dan behavior (dibuka, dibaca, ditutup).
- Komputer memiliki state (merk, RAM, prosesor) dan behavior (menyala, memproses data, mati).



c) Encapsulation

- Encapsulation adalah proses mengikat (binding) data dan metode dalam satu unit (kelas), sekaligus membatasi akses langsung ke data tersebut. Hal ini dilakukan agar data dalam suatu objek tetap aman dan tidak bisa diubah sembarangan dari luar kelas.
- Dengan encapsulation, data tetap aman dan hanya bisa dimanipulasi melalui metode yang telah ditentukan, sehingga mencegah perubahan langsung yang bisa merusak logika program.

d) Inheritance

- *Inheritance* (pewarisan) adalah proses di mana suatu kelas (*child class*) mewarisi properti dan metode dari kelas lain (*parent class*), sehingga memungkinkan penggunaan kembali kode tanpa perlu menulis ulang. Dengan *inheritance*, *child class* dapat menggunakan fitur dari *parent class*, menambahkan fitur baru, atau memodifikasi metode yang ada. Hal ini membuat kode lebih modular, terstruktur, dan mudah dikelola.
- Contohnya, jika class Mobil mewarisi class Kendaraan, maka Mobil akan memiliki semua properti dan metode Kendaraan, seperti merk dan jalan(), sambil tetap bisa menambahkan fitur spesifik seperti klakson() atau jumlahPintu.

e) Abstraction

- Dalam OOP, abstraksi adalah teknik yang menyembunyikan detail implementasi suatu sistem dan hanya menampilkan fungsionalitas utama kepada pengguna. Dalam Java, abstraksi dapat dicapai menggunakan kelas abstrak dan interface.

- Contoh nyata dari abstraksi adalah mobil, di mana pengguna hanya melihat dan menggunakan fitur seperti tombol start, persneling, layar tampilan, dan rem, sementara mekanisme internal seperti cara kerja mesin, proses menyalakan mobil, dan perpindahan gigi tersembunyi. Saat pengguna melakukan suatu aksi pada fitur tersebut, sistem akan menjalankan proses internalnya tanpa harus diketahui oleh pengguna.

f) Polymorphism

- Istilah "polymorphism" berarti "banyak bentuk". Dalam pemrograman berorientasi objek, polimorfisme memungkinkan satu entitas memiliki beberapa bentuk dengan nama yang sama. Di Java, polimorfisme dapat diterapkan melalui dua konsep utama, yaitu method overloading dan method overriding.
- Method overloading terjadi dalam satu kelas, di mana terdapat beberapa metode dengan nama yang sama tetapi memiliki perbedaan dalam jumlah atau tipe parameter. Sementara itu, method overriding terjadi melalui mekanisme pewarisan (inheritance), di mana sebuah metode pada kelas induk dapat ditulis ulang dengan cara yang berbeda dalam kelas turunan, tetap menggunakan nama yang sama.

4.3 OOP vs Procedural Programming

Pemrograman berorientasi objek (OOP) memiliki beberapa keunggulan dibandingkan dengan pemrograman prosedural:

- OOP lebih cepat dan lebih mudah dieksekusi.
- OOP memberikan struktur yang jelas untuk program.
- OOP membantu menjaga kode Java tetap DRY (*Don't Repeat Yourself*), sehingga kode lebih mudah dipelihara, dimodifikasi, dan diperbaiki.
- OOP memungkinkan pembuatan aplikasi yang dapat digunakan kembali secara penuh dengan lebih sedikit kode dan waktu pengembangan yang lebih singkat.

Tips: Prinsip DRY (*Don't Repeat Yourself*) bertujuan untuk mengurangi pengulangan kode. Artinya kita sebaiknya mengekstrak bagian kode yang sering digunakan dalam aplikasi, menyimpannya di satu tempat, dan menggunakannya kembali daripada menulis ulang kode yang sama.

Aspek	**OOP (Object-Oriented Programming)**	**Pemrograman Prosedural**
Pendekatan	Berbasis objek (menggunakan kelas dan objek)	Berbasis prosedur (menggunakan fungsi dan prosedur)
Struktur Program	Terorganisir dalam bentuk kelas dan objek	Terstruktur secara linier dengan serangkaian instruksi
Keunggulan Utama	Reusability, enkapsulasi, dan modularitas	Lebih sederhana dan mudah dipahami untuk program kecil

Fokus Utama	Entitas dunia nyata (objek) dan interaksi antar objek	Urutan eksekusi dan alur logika program
Data & Keamanan	Data terlindungi melalui enkapsulasi (akses terbatas)	Data cenderung terbuka dan bisa diakses dari mana saja
Pemeliharaan & Skalabilitas	Mudah dipelihara dan dikembangkan untuk proyek besar	Bisa sulit dikelola saat program menjadi lebih kompleks
Penggunaan Ulang Kode	Tinggi, karena mendukung inheritance dan polymorphism	Rendah, sering terjadi duplikasi kode
Contoh Bahasa	Java, Python, C++, PHP (OOP)	C, Pascal, Basic, Python (prosedural)

4.4 Implementasi Class dan Object

a) Class

Membuat kelas:

- Menggunakan keyword **class**
- Nama kelas harus diawali dengan huruf besar (PascalCase)
- Jika terdiri dari lebih dari satu kata, gunakan **camel case** (contoh: MahasiswaBaru)
- Nama kelas harus **deskriptif** agar mudah dipahami.
- Contoh benar: Mahasiswa, Mobil, PenggunaAkun
- Contoh salah: mahasiswa, mobil123, pengguna_akun.

Contoh:

```
class Mahasiswa {

}
```

Dalam pemrograman berorientasi objek (OOP), sebuah kelas dapat berisi berbagai elemen yang mendukung fungsionalitasnya. Berikut adalah elemen-elemen yang bisa ada dalam kelas:

1. Atribut (Properties/Fields)
   Atribut adalah variabel yang menyimpan data dalam sebuah kelas. Atribut ini menentukan karakteristik dari objek yang dibuat dari kelas tersebut.
   Contoh:

```java
class Mahasiswa {

    // Atribut
    private String nama;
    public String nim;
    protected int angkatan;
    boolean isLulus = false; // default value

}
```

Ketentuan atribut:

- Menggunakan **camelCase** untuk nama atribut.
- Bisa memiliki **modifier akses** (private, public, protected).
- Bisa memiliki nilai awal (default value)

2. Konstruktor (Constructor)
   Konstruktor adalah metode khusus yang dipanggil saat objek dibuat. Konstruktor digunakan untuk menginisialisasi nilai atribut.
   Contoh:

```java
class Mahasiswa {

    // Atribut
    private String nama;
    public String nim;
    protected int angkatan;
    boolean isLulus = false; // default value

    // Contstructor
    public Mahasiswa(String nama, String nim, int angkatan) {
        this.nama = nama;
        this.nim = nim;
        this.angkatan = angkatan;
    }

}
```

Ketentuan konstruktor:

- Nama konstruktor harus sama dengan nama kelas.
- Tidak memiliki return type (tidak menggunakan void atau tipe lainnya)
- Bisa ada lebih dari satu konstruktor (overloading)

3. Method / Function
   Metode adalah fungsi dalam kelas yang digunakan untuk melakukan aksi atau mengembalikan nilai.
   Contoh:

```java
class Mahasiswa {

    // Atribut
    private String nama;
    public String nim;
    protected int angkatan;
    boolean isLulus = false; // default value

    // Contstructor
    public Mahasiswa(String nama, String nim, int angkatan) {
        this.nama = nama;
        this.nim = nim;
        this.angkatan = angkatan;
    }

    // Method / Function
    public void belajar() {
        System.out.println(this.nama + " sedang belajar");
    }

    public String getNama() {
        return this.nama;
    }

}
```

Ketentuan metode:

- Menggunakan camelCase untuk nama metode.
- Bisa memiliki parameter dan return value
- Bisa menggunakan modifier akses (public, private, dll.)

4. Inner Class (Kelas di dalam kelas)
   Java mendukung kelas dalam kelas lain (inner class) untuk modularitas.
   Contoh:

```java
public class Luar {

    class Dalam {
        void tampilkan() {
            System.out.println(x:"Ini adalah kelas dalam.");
        }
    }

}
```

b) Object

Untuk membuat objek dalam Java, kita menggunakan kata kunci **new** untuk mengalokasikan memori untuk objek baru.
Sintaks dasar:

```
NamaKelas namaObjek = new NamaKelas();
```

Contoh:
Class Mahasiswa: (berperan sebagai blueprint untuk membuat objek)

```
public class Mahasiswa {
    String nama;
    String nim;
    int angkatan;
}
```

Membuat objek dari class Mahasiswa:

```
public class Main {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Mahasiswa mhs1 = new Mahasiswa();
        mhs1.nama = "Budi";
        mhs1.nim = "12345";
        mhs1.angkatan = 2023;
    }
}
```

- Mahasiswa adalah **kelas**.
- mhs1 adalah **objek** dari kelas Mahasiswa

Mengakses / mencetak objek:

```
public class Main {
    Run | Debug
    public static void main(String[] args) throws Exception {
        Mahasiswa mhs1 = new Mahasiswa();
        mhs1.nama = "Budi";
        mhs1.nim = "12345";
        mhs1.angkatan = 2023;

        // Mencetak data mahasiswa dari objek mhs1
        System.out.println("Nama: " + mhs1.nama);
        System.out.println("NIM: " + mhs1.nim);
        System.out.println("Angkatan: " + mhs1.angkatan);

    }
}
```

Output:

```
Nama: Budi
NIM: 12345
Angkatan: 2023
```

Kita juga bisa mengubah data dalam objek:

```java
public class Main {
    Run | Debug
    public static void main(String[] args) throws Exception {

        Mahasiswa mhs1 = new Mahasiswa(nama:"Budi", nim:"12345", angkatan:2023);
        mhs1.profilMahasiswa();

        // mengubah nama mahasiswa
        mhs1.nama = "Andi";

        mhs1.profilMahasiswa();

    }
}
```

4.5 Metode dan Fungsi

a) Metode (Method)

Method adalah fungsi yang dideklarasikan di dalam sebuah kelas dan digunakan untuk menjalankan aksi atau perilaku dari objek. Dengan kata lain, method adalah fungsi yang dimiliki oleh objek dalam OOP.

Contoh Method:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Constructor
    public Mahasiswa(String nama, int umur) {
        this.nama = nama;
        this.umur = umur;
    }

    // Method tanpa parameter
    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    // Method dengan parameter
    public void ubahNama(String namaBaru) {
        this.nama = namaBaru;
    }

    // Method dengan return value
    public int getUmur() {
        return umur;
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa(nama:"Budi", umur:20);
        mhs1.tampilkanInfo(); // Output: Nama: Budi, Umur: 20

        mhs1.ubahNama(namaBaru:"Andi");
        mhs1.tampilkanInfo(); // Output: Nama: Andi, Umur: 20

        System.out.println("Umur: " + mhs1.getUmur()); // Output: Umur: 20
    }
}
```

- **tampilkanInfo()** → Method tanpa parameter, hanya mencetak informasi mahasiswa.
- **ubahNama(String namaBaru)** → Method dengan **parameter** untuk mengubah nilai atribut nama.
- **getUmur()** → Method dengan **return value** untuk mengembalikan umur mahasiswa.

b) Fungsi (Function)

Fungsi adalah sekumpulan kode yang dapat digunakan berulang kali untuk melakukan suatu tugas tertentu. Fungsi biasanya memiliki nama, parameter (opsional), dan nilai kembalian (return value, opsional).

Contoh fungsi:

```java
public class ContohFungsi {
    // Fungsi tanpa parameter dan tanpa return
    public static void sapa() {
        System.out.println(x:"Halo, selamat datang!");
    }

    // Fungsi dengan parameter
    public static void sapaPengguna(String nama) {
        System.out.println("Halo, " + nama + "!");
    }

    // Fungsi dengan parameter dan return value
    public static int tambah(int a, int b) {
        return a + b;
    }

    Run | Debug
    public static void main(String[] args) {
        sapa(); // Output: Halo, selamat datang!
        sapaPengguna(nama:"Budi"); // Output: Halo, Budi!
        System.out.println(tambah(a:5, b:3)); // Output: 8
    }
}
```

- Fungsi sapa() hanya mencetak teks.
- Fungsi sapaPengguna(String nama) menerima **parameter** untuk menyesuaikan output.
- Fungsi tambah(int a, int b) menerima dua angka dan mengembalikan hasil penjumlahan

c) Static Method VS Non-static Method

- Non-static Method: harus dipanggil melalui **objek** karena terkait langsung dengan objek tertentu.

Contoh:

```java
public class Mobil {
    String merk;

    public Mobil(String merk) {
        this.merk = merk;
    }

    public void infoMobil() {
        System.out.println("Mobil ini bermerk: " + merk);
    }

    Run | Debug
    public static void main(String[] args) {
        Mobil mbl1 = new Mobil(merk:"Toyota");
        mbl1.infoMobil(); // Output: Mobil ini bermerk: Toyota
    }
}
```

Method **infoMobil()** hanya bisa dipanggil melalui objek (mbl1).

- Static Method: Static method tidak perlu objek untuk dipanggil, karena milik **kelas**, bukan objek.

Contoh:

```java
public class Kalkulator {
    public static int tambah(int a, int b) {
        return a + b;
    }

    Run | Debug
    public static void main(String[] args) {
        System.out.println(Kalkulator.tambah(a:10, b:5)); // Output: 15
    }
}
```

tambah() bisa langsung dipanggil tanpa membuat objek (Kalkulator.tambah(10, 5)).

d) Method Overloading
Adalah method dengan **nama yang sama tetapi parameter berbeda**.

Contoh:

```java
public class Hitung {
    public int tambah(int a, int b) {
        return a + b;
    }

    public int tambah(int a, int b, int c) {
        return a + b + c;
    }

    Run | Debug
    public static void main(String[] args) {
        Hitung h = new Hitung();
        System.out.println(h.tambah(a:3, b:5)); // Output: 8
        System.out.println(h.tambah(a:3, b:5, c:2)); // Output: 10
    }
}
```

Java akan memilih method sesuai dengan jumlah parameter yang diberikan.

4.6 Konstruktor

Konstruktor adalah method khusus dalam sebuah kelas yang digunakan untuk menginisialisasi objek saat objek tersebut dibuat. Konstruktor dipanggil secara otomatis ketika objek baru dibuat.

Karateristik konstruktor:

- Nama konstruktor harus sama dengan nama kelasnya.
- Tidak memiliki tipe data pengembalian (return type), bahkan void pun tidak boleh.
- Dipanggil secara otomatis saat objek dibuat.
- Digunakan untuk menginisialisasi atribut objek.

a) Jenis-jenis konstruktor dalam Java

Dalam Java, terdapat tiga jenis konstruktor utama:

1. Konstruktor default (tanpa parameter)
   Konstruktor ini otomatis dibuat oleh Java jika tidak ada konstruktor yang dideklarasikan dalam kelas.

   Contoh:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Konstruktor default (tanpa parameter)
    public Mahasiswa() {
        System.out.println(x:"Konstruktor default dipanggil!");
        nama = "Tidak diketahui";
        umur = 0;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa(); // Konstruktor dipanggil otomatis
        mhs1.tampilkanInfo(); // Output: Nama: Tidak diketahui, Umur: 0
    }
}
```

- Konstruktor Mahasiswa() dipanggil otomatis saat objek mhs1 dibuat.
- Atribut nama dan umur diinisialisasi dengan nilai default.

2. Konstruktor Berparameter
   Konstruktor yang menerima **parameter** untuk menginisialisasi objek dengan nilai yang diberikan saat objek dibuat.

   Contoh:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Konstruktor dengan parameter
    public Mahasiswa(String nama, int umur) {
        this.nama = nama;
        this.umur = umur;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa(nama:"Budi", umur:20);
        mhs1.tampilkanInfo(); // Output: Nama: Budi, Umur: 20

        Mahasiswa mhs2 = new Mahasiswa(nama:"Andi", umur:22);
        mhs2.tampilkanInfo(); // Output: Nama: Andi, Umur: 22
    }
}
```

- Konstruktor menerima parameter nama dan umur untuk menginisialisasi objek.
- **Keyword this** digunakan untuk membedakan antara variabel instance dan parameter.

3. Konstructor Copy (Copy constructor)
   Adalah konstruktor yang digunakan untuk membuat objek baru dengan **menyalin nilai dari objek lain**.

   Contoh:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Konstruktor dengan parameter
    public Mahasiswa(String nama, int umur) {
        this.nama = nama;
        this.umur = umur;
    }

    // Konstruktor Copy
    public Mahasiswa(Mahasiswa m) {
        this.nama = m.nama;
        this.umur = m.umur;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa(nama:"Budi", umur:20);
        Mahasiswa mhs2 = new Mahasiswa(mhs1); // Salin dari objek mhs1

        mhs1.tampilkanInfo(); // Output: Nama: Budi, Umur: 20
        mhs2.tampilkanInfo(); // Output: Nama: Budi, Umur: 20
    }
}
```

   Konstruktor **copy** digunakan untuk menyalin atribut dari objek mhs1 ke objek mhs2.

4. Overloading Konstruktor
   Java mendukung **overloading konstruktor**, yaitu membuat beberapa konstruktor dalam satu kelas tetapi dengan parameter berbeda.

   Contoh:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Konstruktor 1: Default
    public Mahasiswa() {
        this.nama = "Tidak diketahui";
        this.umur = 0;
    }

    // Konstruktor 2: Dengan satu parameter
    public Mahasiswa(String nama) {
        this.nama = nama;
        this.umur = 18; // Default umur
    }

    // Konstruktor 3: Dengan dua parameter
    public Mahasiswa(String nama, int umur) {
        this.nama = nama;
        this.umur = umur;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa();
        Mahasiswa mhs2 = new Mahasiswa(nama:"Budi");
        Mahasiswa mhs3 = new Mahasiswa(nama:"Andi", umur:21);

        mhs1.tampilkanInfo(); // Output: Nama: Tidak diketahui, Umur: 0
        mhs2.tampilkanInfo(); // Output: Nama: Budi, Umur: 18
        mhs3.tampilkanInfo(); // Output: Nama: Andi, Umur: 21
    }
}
```

Java memilih konstruktor berdasarkan parameter yang diberikan saat pembuatan objek.

5. Keyword this() dalam konstruktor
   this() digunakan untuk **memanggil konstruktor lain** dalam kelas yang sama untuk menghindari kode yang duplikatif.

   Contoh:

```java
public class Mahasiswa {
    String nama;
    int umur;

    // Konstruktor utama
    public Mahasiswa() {
        this(nama:"Tidak diketahui", umur:0); // Memanggil konstruktor lain
    }

    public Mahasiswa(String nama) {
        this(nama, umur:18); // Default umur
    }

    public Mahasiswa(String nama, int umur) {
        this.nama = nama;
        this.umur = umur;
    }

    public void tampilkanInfo() {
        System.out.println("Nama: " + nama + ", Umur: " + umur);
    }

    Run | Debug
    public static void main(String[] args) {
        Mahasiswa mhs1 = new Mahasiswa();
        Mahasiswa mhs2 = new Mahasiswa(nama:"Budi");
        Mahasiswa mhs3 = new Mahasiswa(nama:"Andi", umur:21);

        mhs1.tampilkanInfo(); // Output: Nama: Tidak diketahui, Umur: 0
        mhs2.tampilkanInfo(); // Output: Nama: Budi, Umur: 18
        mhs3.tampilkanInfo(); // Output: Nama: Andi, Umur: 21
    }
}
```

this("Tidak diketahui", 0); memanggil konstruktor lain dengan parameter.

b) Perbedaan Konstruktor dan Method

Perbedaan	Konstruktor	Method
Nama	Sama dengan nama kelas	Bisa diberi nama apa saja
Return Type	Tidak memiliki return type	Bisa memiliki return type (void, int, String, dll.)
Pemanggilan	Dipanggil otomatis saat objek dibuat	Dipanggil secara eksplisit oleh objek
Tujuan	Menginisialisasi objek	Melakukan operasi tertentu

Latihan

Kerjakan soal dibawah ini!

Latihan	
Durasi Pengerjaan	30 menit
Start	16:10
End	18:00
Soal	**Sistem Manajemen Mahasiswa** Buatlah sebuah program Java untuk mengelola data mahasiswa. Program harus memiliki class Mahasiswa dengan atribut: • nama (String) • nim (String) • jurusan (String) • ipk (double) Buatlah fitur berikut: 1. **Constructor** untuk menginisialisasi data mahasiswa. 2. **Method** tampilkanInfo() yang mencetak informasi mahasiswa. 3. **Method** cekLulus() yang mengembalikan **true** jika ipk >= 3.0, jika tidak **false**. **Instruksi:** 1. Buat class Mahasiswa sesuai spesifikasi di atas. 2. Buat Main class untuk membuat objek dari Mahasiswa dan uji method yang dibuat.
Link Pengumpulan	https://forms.gle/ebjyBbCyQggqtdDT6

Posttest

Kerjakan soal dibawah ini!

Posttest	
Durasi Pengerjaan	30 menit
Start	16:10
End	23:59
Soal	Sebuah perusahaan penyewaan kendaraan ingin mengembangkan sistem manajemen penyewaan mobil menggunakan Java. Sistem ini harus dapat mencatat informasi setiap mobil yang tersedia,

	seperti nomor plat, merek, harga sewa per hari, dan status ketersediaan mobil tersebut. Untuk memungkinkan fleksibilitas dalam pembuatan objek, sistem harus memiliki dua jenis konstruktor pada kelas Mobil, yaitu konstruktor default yang menginisialisasi atribut dengan nilai bawaan, serta konstruktor berparameter yang memungkinkan pengisian data mobil secara langsung. Mobil juga harus memiliki metode untuk menampilkan informasi serta mengubah status ketersediaan setelah disewa. Setiap pelanggan yang ingin menyewa mobil harus memiliki informasi berupa nama, nomor KTP, dan nomor HP, yang akan disimpan dalam objek pelanggan. Pelanggan juga harus memiliki metode untuk menampilkan informasi pribadinya. Untuk menangani transaksi penyewaan, sistem memerlukan kelas Sewa yang menyimpan data pelanggan, mobil yang disewa, serta durasi penyewaan dalam hitungan hari. Saat transaksi dibuat, sistem harus secara otomatis menghitung total biaya berdasarkan harga sewa mobil dan lama sewa. Transaksi harus memiliki metode untuk memproses penyewaan, di mana sistem akan memeriksa apakah mobil tersedia. Jika tersedia, maka status mobil harus berubah menjadi tidak tersedia, dan sistem akan menampilkan struk penyewaan. Jika mobil tidak tersedia, maka transaksi gagal. Struk penyewaan harus mencantumkan informasi pelanggan, mobil yang disewa, lama sewa, total biaya, serta status transaksi. Selain itu, sistem harus memiliki kelas tambahan Utility yang berisi metode static untuk menghitung diskon. Jika pelanggan menyewa mobil selama lebih dari 5 hari, maka mereka akan mendapatkan diskon sebesar 10% dari total biaya sewa. Kelas Utility juga harus menyediakan metode untuk mengonversi angka ke dalam format mata uang yang lebih mudah dibaca. Tugas Anda adalah mengimplementasikan sistem ini dalam bahasa Java dengan menerapkan enkapsulasi menggunakan modifier private pada atribut serta membuat getter dan setter jika diperlukan. Gunakan method overloading dalam konstruktor kelas Mobil, serta manfaatkan this() untuk menghindari duplikasi kode dalam konstruktor. Pastikan untuk menggunakan static method dalam kelas Utility untuk perhitungan diskon. Terakhir, buat kelas Main untuk menjalankan program dengan skenario berikut: membuat beberapa objek Mobil dan Pelanggan, menampilkan daftar mobil yang tersedia, melakukan transaksi penyewaan, menampilkan struk penyewaan, serta memperbarui status mobil setelah transaksi.
Link Pengumpulan	https://forms.gle/TLhuWu5byMGs5UEK9

Take Home Task

Kerjakan soal dibawah ini!

Take Home	
Durasi Pengerjaan	6 hari
Start	Kamis
End	Rabu, 26 Maret 2025, pukul 23:59
Template laporan	contoh template laporan tht
Soal	**Studi Kasus: Sistem Manajemen Toko Online** **Deskripsi Kasus:** Sebuah toko online ingin mengembangkan sistem untuk mengelola produk, pelanggan, dan transaksi. Sistem ini harus memiliki fitur berikut: 1. **Manajemen Produk** • Setiap produk memiliki kode unik, nama, harga, dan stok. • Produk dapat ditambahkan, diperbarui, dan ditampilkan informasinya. 2. **Manajemen Pelanggan** • Setiap pelanggan memiliki ID unik, nama, email, dan saldo akun. • Pelanggan dapat melakukan top-up saldo. 3. **Manajemen Transaksi** • Pelanggan dapat membeli produk jika saldo mencukupi. • Saat transaksi berhasil, saldo pelanggan berkurang, dan stok produk berkurang. • Jika saldo tidak mencukupi atau stok habis, transaksi gagal. • Riwayat transaksi harus disimpan. **Struktur Kelas & Fitur yang Harus Ada:** **1. Class Produk** Atribut: • kodeProduk (String)

	• namaProduk (String) • harga (double) • stok (int) Method: • **Constructor** untuk inisialisasi data produk. • **Method tampilkanInfoProduk()** → Menampilkan detail produk. • **Method kurangiStok(int jumlah)** → Mengurangi stok jika tersedia.
	2. Class Pelanggan Atribut: • idPelanggan (String) • nama (String) • email (String) • saldo (double) Method: • **Constructor** untuk inisialisasi data pelanggan. • **Method tampilkanInfoPelanggan()** → Menampilkan informasi pelanggan. • **Method topUpSaldo(double jumlah)** → Menambah saldo pelanggan. • **Method kurangiSaldo(double jumlah)** → Mengurangi saldo saat transaksi.
	3. Class Transaksi Atribut: • idTransaksi (String) • pelanggan (Pelanggan) • produk (Produk) • jumlahBeli (int) • totalHarga (double)

	Method:
	• **Constructor** untuk membuat transaksi.
	• **Method prosesTransaksi()**
	  ○ Mengecek apakah saldo pelanggan cukup.
	  ○ Mengecek apakah stok produk tersedia.
	  ○ Jika berhasil, stok produk berkurang dan saldo pelanggan berkurang.
	  ○ Jika gagal, tampilkan pesan error.
	• **Method tampilkanDetailTransaksi() →** Menampilkan riwayat transaksi.

Link Pengumpulan	