

MODUL 10

ABSTRACT & INTERFACE

10.1. Abstract

- Dalam dunia OOP (Object-Oriented Programming), abstraksi adalah proses untuk menyembunyikan detail yang tidak penting, dan hanya menampilkan fitur penting dari suatu objek kepada pengguna.
- Contoh sehari-hari:
Saat kita menyalakan lampu, kita hanya menekan saklar. Kita tidak perlu tahu bagaimana aliran listrik bekerja.
Itulah abstraksi: kita hanya berinteraksi dengan fitur penting (saklar), bukan detail teknis (kabel, listrik, dll). Tujuan utama abstraction adalah menyederhanakan kompleksitas sistem dan hanya menampilkan hal-hal yang penting dari sebuah objek.
- Java menyediakan dua cara utama untuk mengimplementasikan abstraksi yakni melalui *abstract class* dan *Interface*.

1). Abstract Class

- *abstract class* adalah kelas dasar yang tidak bisa langsung dibuat objeknya. Biasanya digunakan sebagai kerangka dasar untuk kelas-kelas lain (subclass).
- Dapat memiliki method abstrak (tanpa implementasi) dan method biasa (dengan implementasi).
- Sintaks dasar:

```
abstract class Hewan {  
  
}
```

Cara membuat abstract class:

- 1). Gunakan keyword `abstract` sebelum class
- 2). Di dalamnya bisa ada:
 - method abstrak (tanpa isi)
 - method biasa (dengan isi)
 - variabel
 - constructor (boleh)

Contoh:

```

Main.java

// Abstract class
abstract class Hewan {
    // variabel
    String nama;

    // constructor
    Hewan(String nama) {
        this.nama = nama;
    }

    // method biasa
    void makan() {
        System.out.println(nama + " sedang makan.");
    }

    // method abstract
    abstract void bersuara();
}

```

Cara menggunakannya adalah dengan membuat subclass (anak kelas) dan override method abstract:

```

Main.java

class Kucing extends Hewan {
    Kucing(String nama) {
        super(nama);
    }

    @Override
    void bersuara() {
        System.out.println(nama + " berkata: Meong!");
    }
}

public class Main {
    public static void main(String[] args) {
        Kucing k = new Kucing("Kitty");
        k.makan();           // Output: Kitty sedang makan.
        k.bersuara();        // Output: Kitty berkata: Meong!
    }
}

```

➤ **Apa bedanya abstract class dan abstract method?**

abstract class dan abstract method adalah dua hal yang berbeda namun berhubungan dalam konsep abstraksi di Java.

Aspek	abstract class	abstract method
Apa itu?	Sebuah kelas yang tidak lengkap	Sebuah method tanpa isi (body)
Keyword	abstract class	Ditulis dengan keyword abstract dalam method
Tujuan	Sebagai kerangka dasar untuk subclass	Memaksa subclass untuk memberikan implementasi
Isi	Bisa punya method biasa & abstract method	Tidak punya isi, hanya deklarasi
Wajib override?	Tidak selalu semua method-nya abstract	Harus di- override di subclass
Contoh?	abstract class Hewan {}	abstract void suara();

Contoh:

- abstract class dengan abstract method:

```
Hewan.java

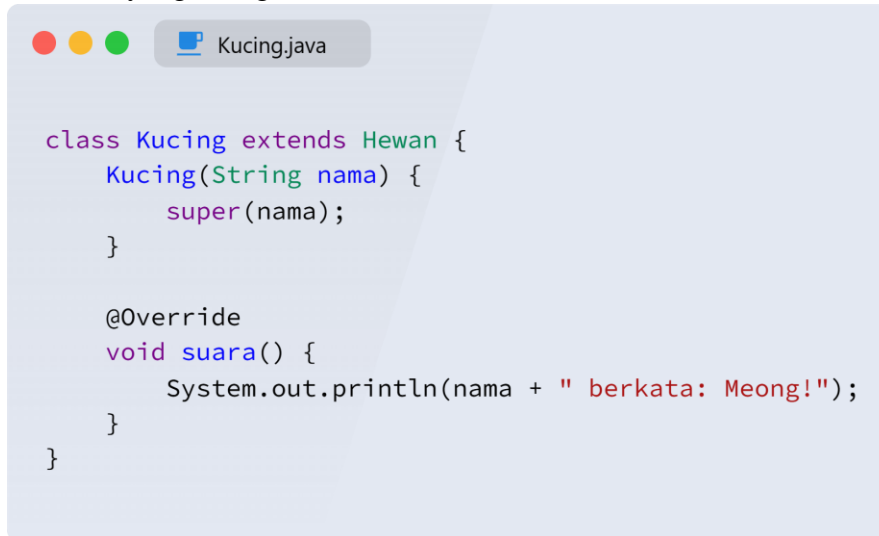
abstract class Hewan {
    String nama;

    // Constructor
    Hewan(String nama) {
        this.nama = nama;
    }

    // Method biasa
    void makan() {
        System.out.println(nama + " sedang makan.");
    }

    // Method abstract (harus di-override oleh subclass)
    abstract void suara();
}
```

- Subclass yang Meng-Override abstract method:



```
class Kucing extends Hewan {
    Kucing(String nama) {
        super(nama);
    }

    @Override
    void suara() {
        System.out.println(nama + " berkata: Meong!");
    }
}
```

- Main method:



```
public class Main {
    public static void main(String[] args) {
        Hewan h = new Kucing("Kitty");
        h.makan(); // Output: Kitty sedang makan.
        h.suara(); // Output: Kitty berkata: Meong!
    }
}
```

abstract method hanya boleh ada di abstract class atau interface. Jadi Jika kita punya abstract method, maka class tempatnya berada harus jadi abstract juga.

➤ **Ringkasan:**

- abstract class: kerangka dasar yang bisa berisi method normal dan abstract method.
- abstract method: hanya deklarasi method saja, harus diimplementasi oleh subclass.
- Jika sebuah kelas memiliki setidaknya satu method abstrak, maka kelas tersebut harus dideklarasikan sebagai abstract.
- Subclass yang mewarisi abstract class harus mengimplementasikan semua method abstrak (mengoverride).
- Kita tidak bisa membuat object langsung dari abstract class.

10.2. Interface

interface adalah kontrak yang menyatakan bahwa kelas yang mengimplementasikannya harus menyediakan implementasi untuk semua method di dalam interface tersebut.

Karakteristik Interface:

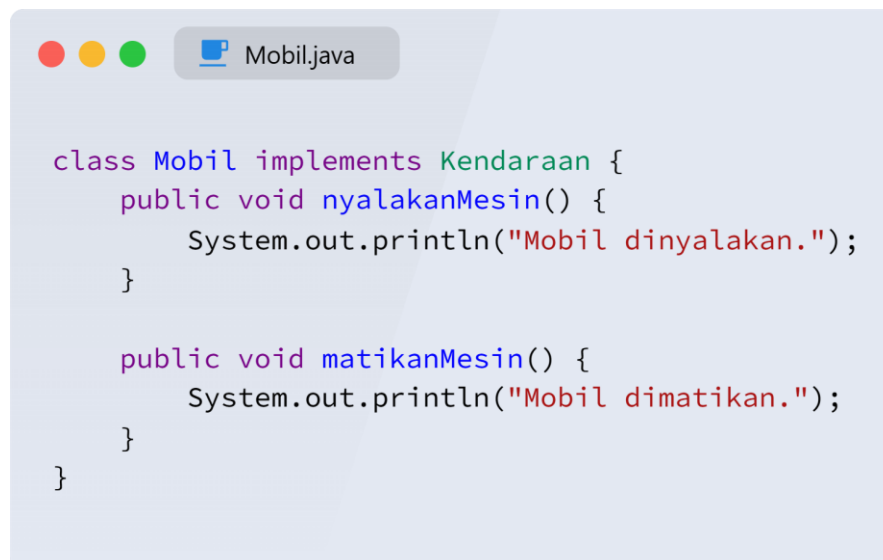
- Menggunakan keyword interface
- Method dalam interface secara default bersifat public dan abstract.
- Interface tidak bisa memiliki constructor.
- Interface tidak bisa membuat objek langsung.
- Satu class bisa mengimplementasikan lebih dari satu interface (Multiple Inheritance).
- Sejak Java 8, interface juga bisa memiliki:
 - default methods (dengan isi),
 - static methods (dengan isi)

Cara Membuat Interface:



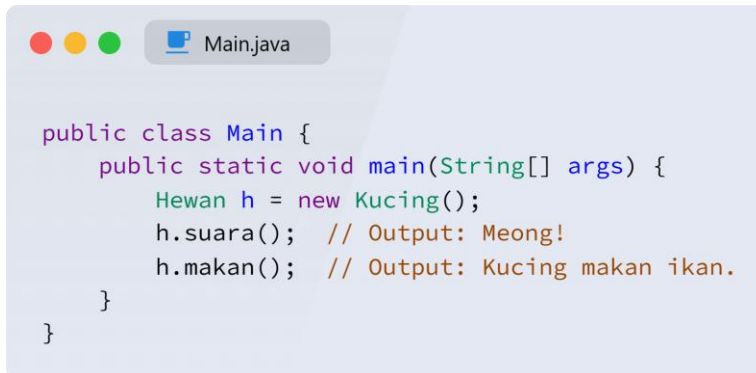
```
interface Kendaraan {  
    void nyalakanMesin(); // abstract method (tanpa isi)  
    void matikanMesin();  
}
```

Cara Mengimplementasikan Interface (menggunakan keyword implements):



```
class Mobil implements Kendaraan {  
    public void nyalakanMesin() {  
        System.out.println("Mobil dinyalakan.");  
    }  
  
    public void matikanMesin() {  
        System.out.println("Mobil dimatikan.");  
    }  
}
```

Main method:



```
public class Main {
    public static void main(String[] args) {
        Hewan h = new Kucing();
        h.suara(); // Output: Meong!
        h.makan(); // Output: Kucing makan ikan.
    }
}
```

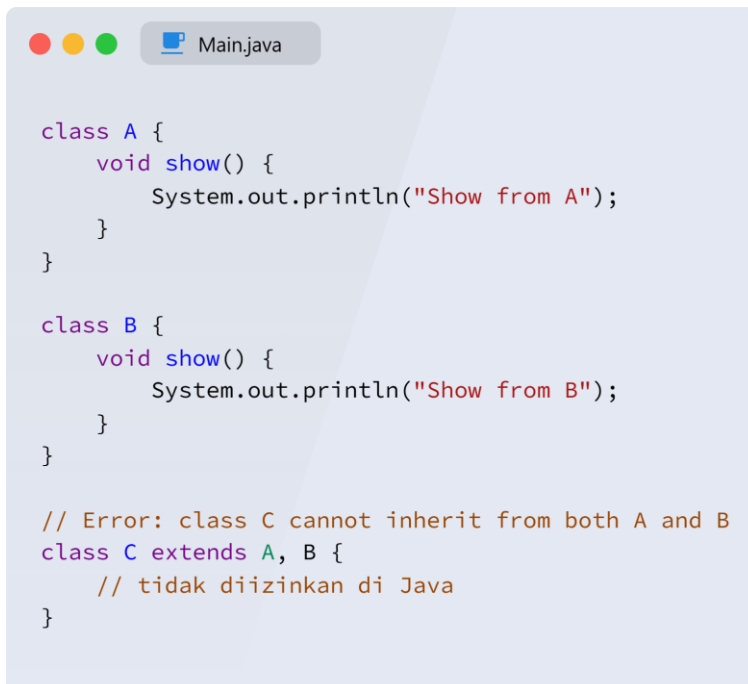
Java tidak mendukung pewarisan ganda dengan class karena bisa menyebabkan konflik. Pewarisan ganda (multiple inheritance) adalah konsep dalam pemrograman berorientasi objek di mana sebuah kelas dapat mewarisi fitur (metode atau atribut) dari lebih dari satu superclass (kelas induk).

Misalnya, jika:

- Kelas A memiliki metode methodA()
- Kelas B memiliki metode methodB()
- Dan kita ingin membuat kelas C yang mewarisi baik dari A maupun B

Hal ini disebut **multiple inheritance**.

Java **tidak mendukung** pewarisan ganda dengan **class** karena bisa menyebabkan konflik. Contohnya:



```
class A {
    void show() {
        System.out.println("Show from A");
    }
}

class B {
    void show() {
        System.out.println("Show from B");
    }
}

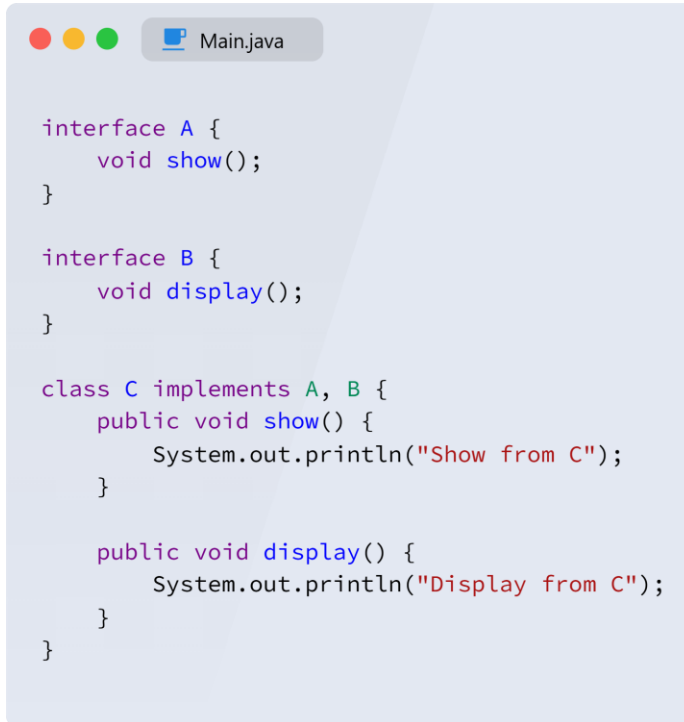
// Error: class C cannot inherit from both A and B
class C extends A, B {
    // tidak diizinkan di Java
}
```

Masalahnya: Jika C memanggil show(), mana yang akan dipanggil? dari A atau B? Inilah yang disebut diamond problem.

Untuk menghindari konflik, Java memungkinkan pewarisan ganda melalui **interface**, karena:

- Interface hanya mendeklarasikan metode (sebelum Java 8).
- Java mengharuskan kelas untuk mengimplementasikan sendiri metode tersebut (atau override default-nya).

Contoh:



```
interface A {  
    void show();  
}  
  
interface B {  
    void display();  
}  
  
class C implements A, B {  
    public void show() {  
        System.out.println("Show from C");  
    }  
  
    public void display() {  
        System.out.println("Display from C");  
    }  
}
```

Di sini C mewarisi dari dua interface, dan tidak terjadi konflik.

Perbandingan singkat:

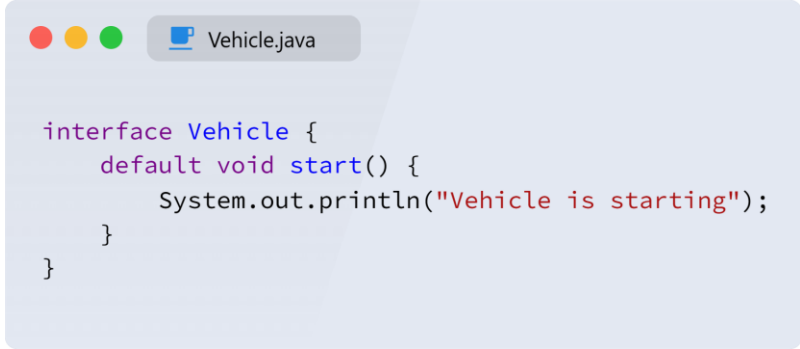
Fitur	Class	Interface
Pewarisan Ganda	Tidak Bisa	Bisa
Kata Kunci	extends	implements
Warisan Atribut & Method	Ya	Hanya method (tanpa implementasi, sebelum Java 8)
Solusi Conflict	Tidak jelas (diamond problem)	Harus override (lebih jelas)

Interface modern (Java 8++)

Interface Modern di Java 8+ membawa peningkatan besar dibanding versi sebelumnya. Berikut adalah beberapa fitur baru di interface modern:

1) Default Method

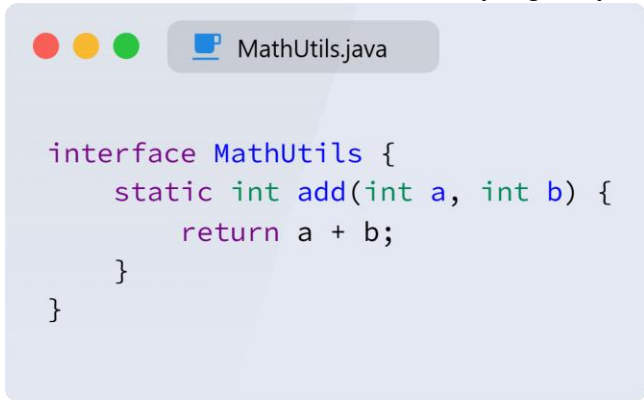
Interface sekarang bisa punya implementasi metode!



```
interface Vehicle {  
    default void start() {  
        System.out.println("Vehicle is starting");  
    }  
}
```

2) Static Method

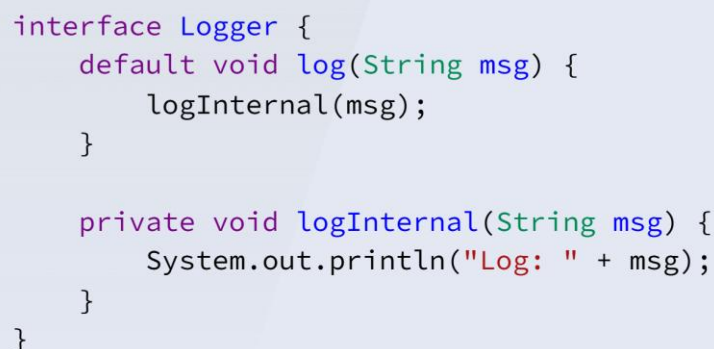
Interface bisa memiliki metode static yang hanya bisa diakses lewat nama interface.



```
interface MathUtils {  
    static int add(int a, int b) {  
        return a + b;  
    }  
}
```

3) Private Method (Java 9+)

Untuk membantu mengelola kode di dalam interface (tidak bisa diakses dari luar interface).



```
interface Logger {  
    default void log(String msg) {  
        logInternal(msg);  
    }  
  
    private void logInternal(String msg) {  
        System.out.println("Log: " + msg);  
    }  
}
```


Contoh lengkap implementasi dengan java modern:

```
Gadget.java

interface Gadget {
    void hidupkan();

    // Default method
    default void cekBaterai() {
        System.out.println("Baterai 100%");
    }

    // Static method
    static void info() {
        System.out.println("Ini adalah interface Gadget.");
    }
}

Smartphone.java

class Smartphone implements Gadget {
    public void hidupkan() {
        System.out.println("Smartphone menyala.");
    }
}
```

Kenapa menggunakan interface?

Alasan	Penjelasan
Multiple Inheritance	Java tidak mendukung pewarisan ganda dengan class, tapi bisa dengan interface.
Loose Coupling	Class hanya tergantung pada kontrak, bukan implementasi konkret.
Polymorphism	Memungkinkan berbagai class menggunakan interface yang sama dengan cara berbeda.
Standardisasi	Menyediakan pola baku bagi banyak class yang berbeda.

Contoh penggunaan abstract dan interface:

```
interface Terbang {
    void terbang();
}

abstract class Burung {
    public abstract void suara();
    public void tidur() {
        System.out.println("Burung tidur di sarang.");
    }
}

class Elang extends Burung implements Terbang {
    public void suara() {
        System.out.println("Sreeek!");
    }

    public void terbang() {
        System.out.println("Elang terbang tinggi.");
    }
}
```

Perbedaan Abstract Class vs Interface:

Fitur	Abstract Class	Interface
Keyword	abstract class	interface
Object	Tidak bisa diinstansiasi	Tidak bisa diinstansiasi
Method	Bisa abstrak dan non-abstrak	Hanya abstrak (sebelum Java 8)
Field	Bisa field normal	Hanya public static final
Inheritance	Hanya bisa extend 1 class	Bisa implement banyak interface
Constructor	Bisa punya constructor	Tidak bisa

10.3. Kapan Menggunakan Abstrac dan Interface?

Gunakan abstract class jika:

- Kita ingin menyediakan implementasi default
- Kita ingin berbagi code antar subclass

Gunakan interface jika:

- Kita ingin menentukan kontrak umum untuk banyak kelas
- Kita butuh multiple inheritance

Take Home Task

Kerjakan soal dibawah ini!

Tugas	
Durasi Pengerjaan	6 hari
Start	
End	Rabu, 11 Juni 2025, jam 23.59
Template laporan	contoh template laporan tht
Soal	<p>SISTEM SEWA KENDARAAN</p> <p>1. Buat abstract class Kendaraan yang memiliki:</p> <ul style="list-style-type: none">• Atribut: <i>platNomor</i>, <i>merk</i>, <i>tahunProduksi</i>• Constructor untuk inisialisasi atribut• Method:<ul style="list-style-type: none">- <i>tampilkanInfo()</i> – method konkrit untuk menampilkan informasi kendaraan- <i>hitungBiayaSewa(int hari)</i> – abstract method- <i>perluSupir()</i> – abstract method <p>2. Buat dua interface:</p> <ul style="list-style-type: none">• <i>DapatDisewa</i> yang memiliki method:<ul style="list-style-type: none">- <i>hitungBiayaSewa(int hari)</i>- <i>perluSupir()</i>• <i>Muatan</i> yang memiliki method:<ul style="list-style-type: none">- <i>kapasitasMuatan()</i> mengembalikan double (dalam kg) <p>3. Buat minimal 3 class turunan:</p> <ul style="list-style-type: none">• <i>MobilPribadi</i> (extends <i>Kendaraan</i>, implements <i>DapatDisewa</i>)• <i>Bus</i> (extends <i>Kendaraan</i>, implements <i>DapatDisewa</i>)• <i>Truk</i> (extends <i>Kendaraan</i>, implements <i>DapatDisewa</i>, <i>Muatan</i>) <p>4. Buat kelas Main untuk:</p> <ul style="list-style-type: none">• Coba semua implementasi nya
Link Pengumpulan	https://forms.gle/VChZvPXeqcSxs9fa9