Final

- Final class → tidak dapat di-extend
- Final method → tidak dapat di-override
- Final attribute → nilainya tidak dapat diubah

Keyword Static

- Instance atribut dan method → dimiliki dan dilakukan oleh object
- Static attribute dan method:
  - atribut dan method yang dimiliki suatu class, bukan objek tertentu dari class tersebut
  - atribut dan method yang dapat diakses/dipanggil tanpa instansiasi objek terlebih dahulu
- Static/Class attribute digunakan jika:
  - Nilai variable independen terhadap object
- Static/Class method digunakan jika:
  - Tidak perlu mengakses instance attribute atau memanggil instance method
  - Membuat fungsi utilitas/helper/service

Contoh 1

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

    public static int multiply(int a, int b) {
        return a * b;
    }
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        int result = Calculation.add(5, 10);
        System.out.println(result);
    }
}
```

Contoh 2

```java
public class Config {
    public static String APP_NAME = "Sistem Akademik";
    public static String VERSION = "1.0";
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(Config.APP_NAME);
        System.out.println(Config.VERSION);
    }
}
```

Contoh 3 – Collections & String

```
ArrayList<String> daftarNama = new ArrayList<>();
daftarNama.add("Cica");
daftarNama.add("Ani");
daftarNama.add("Budi");

Collections.sort(daftarNama);
System.out.println(String.join(", ", daftarNama));
```

```
Ani, Budi, Cica
```

Contoh 4 – Math & Random

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(Math.pow(2, 5));
        System.out.println(Math.random());

        Random random = new Random();
        System.out.println(random.nextInt(100));
    }
}
```

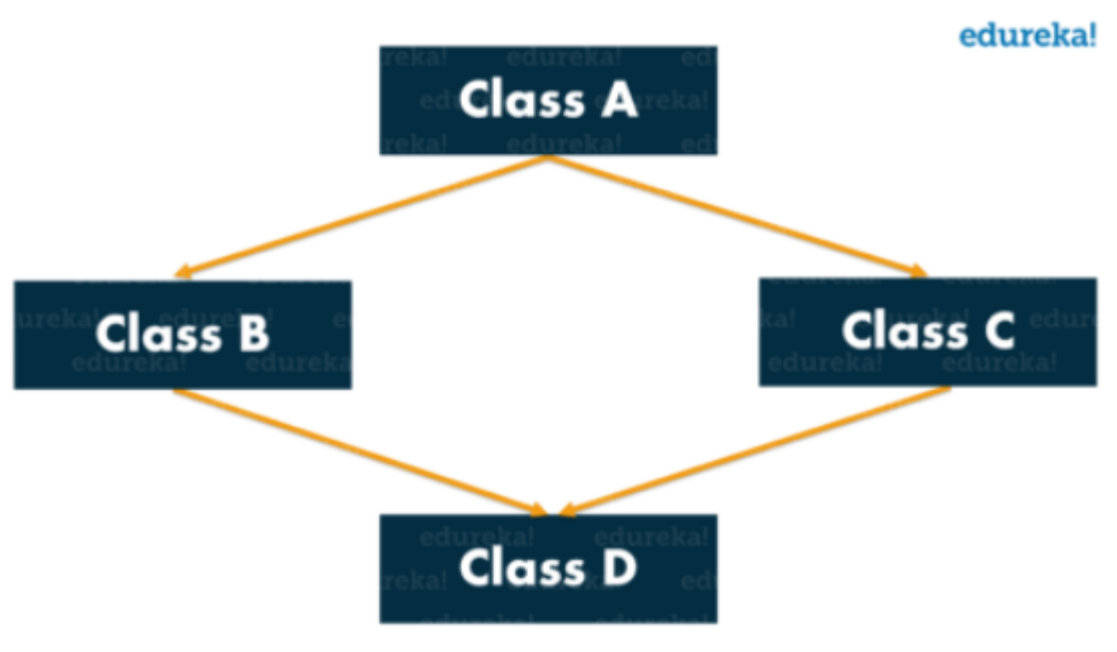
```
32.0
0.9640205110585467
5
```

Contoh 5

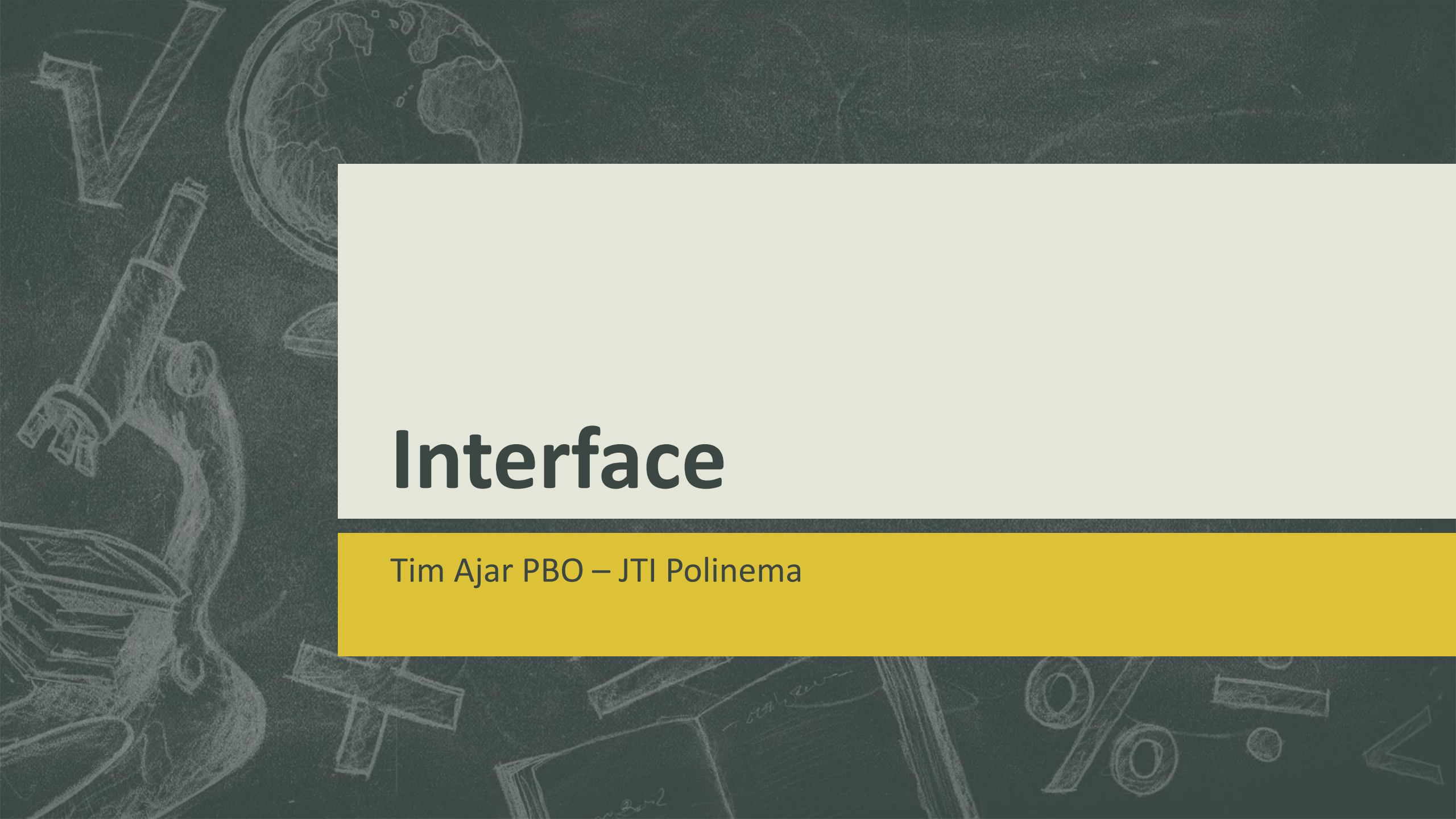
```
public class Lingkaran {
    public static final double phi = 3.14;
    public double jariJari;

    public double getLuas(){
        return phi * jariJari * jariJari;
    }

    public static void welcome(){
        System.out.println("Selamat datang...");
    }
}
```

```
Lingkaran lingkaran1 = new Lingkaran();
lingkaran1.jariJari = 10;
System.out.println(lingkaran1.getLuas());

System.out.println(Lingkaran.phi);
Lingkaran.welcome();
Lingkaran.phi = 4;  //cannot assign value to a final variable
```

Multiple Inheritance



- Multiple inheritance: satu kelas turunan mencoba untuk **memperluas (extend) lebih dari satu kelas induk**.
- Misal terdapat method *show()* pada kelas B dan C dengan fungsi yang berbeda. Kemudian kelas A meng-extend kelas B dan C. Ketika objek dari kelas A mencoba memanggil method *show()*, Java compiler akan bingung method di kelas mana yang akan dieksekusi (dari kelas B atau C)
- Mengarah pada ambiguitas.



Interface

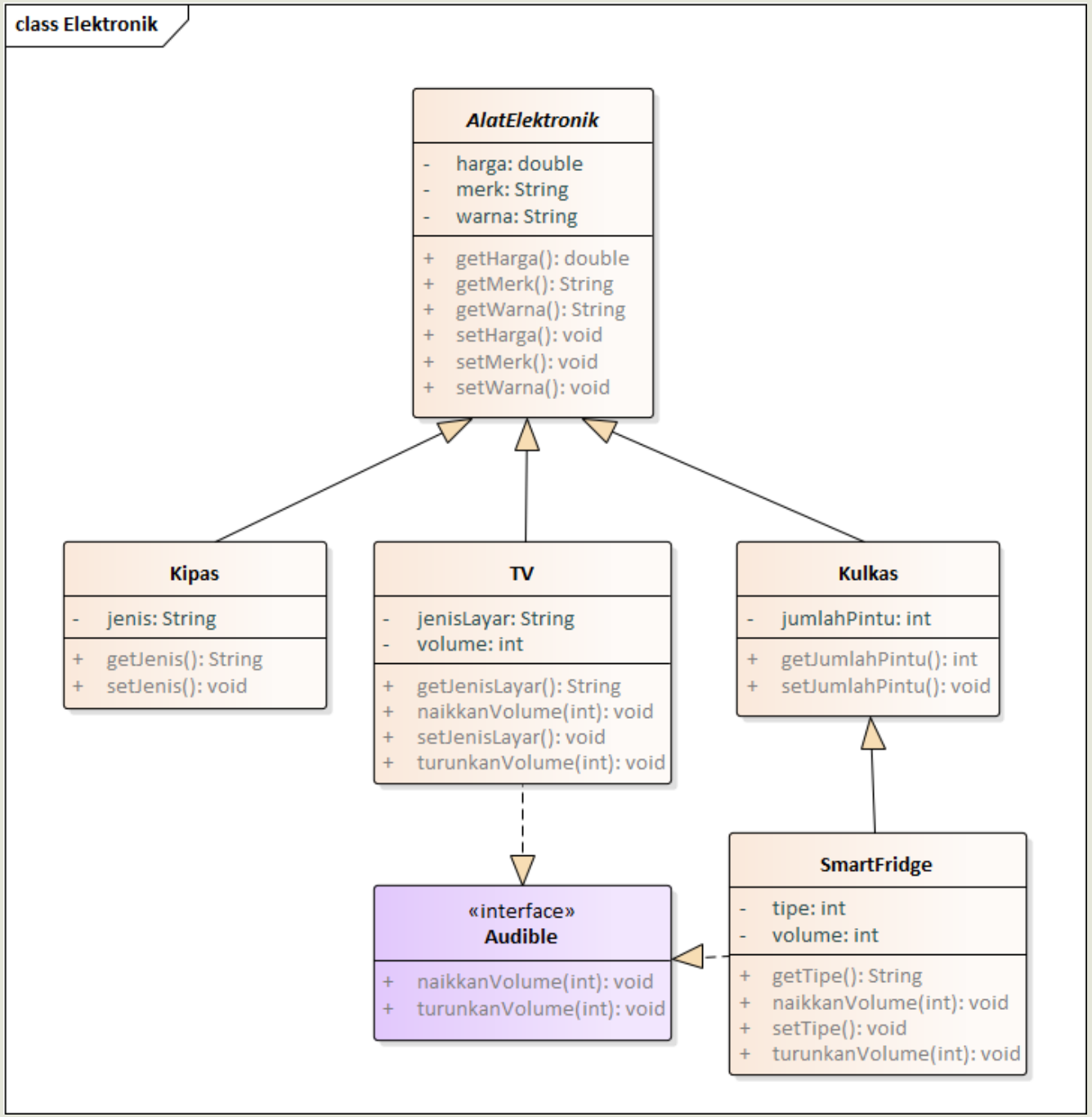
Tim Ajar PBO – JTI Polinema

Definisi Interface

- Interface adalah sekumpulan abstract method yang saling berkaitan
- Bertindak sebagai **kontrak/syarat** yang berisi sekumpulan *behavior/method* yang saling **terkait** untuk memenuhi suatu **kapabilitas**
- Dengan kata lain, interface memberikan panduan mengenai method apa saja yang perlu diimplementasikan untuk memenuhi kapabilitas tertentu
- Karakteristik:
  - Umumnya hanya terdiri dari *abstract methods*
  - Tidak memiliki constructor dan tidak dapat diinstansiasi. Yang dapat diinstasiasi adalah class yang meng-***implement*** interface tersebut
  - Dapat memiliki atribut tetapi hanya bersifat **public static final**

Notasi Class Diagram

- Nama interface **tidak** dicetak miring
- Keterangan **<<interface>>** di atas nama interface
- Nama method dicetak miring atau tidak
- **Implements** dilambangkan dengan garis panah putus-putus


class Elektronik
AlatElektronik
- harga: double
- merk: String
- warna: String
+ getHarga(): double
+ getMerk(): String
+ getWarna(): String
+ setHarga(): void
+ setMerk(): void
+ setWarna(): void
Kipas
- jenis: String
+ getJenis(): String
+ setJenis(): void
TV
- jenisLayar: String
- volume: int
+ getJenisLayar(): String
+ naikkanVolume(int): void
+ setJenisLayar(): void
+ turunkanVolume(int): void
Kulkas
- jumlahPintu: int
+ getJumlahPintu(): int
+ setJumlahPintu(): void
«interface»
Audible
+ naikkanVolume(int): void
+ turunkanVolume(int): void
SmartFridge
- tipe: int
- volume: int
+ getTipe(): String
+ naikkanVolume(int): void
+ setTipe(): void
+ turunkanVolume(int): void

Sintaks Interface

- Untuk mendeklarasikan suatu interface:
  - **`public interface`** `<NamaInterface>`
- Untuk mengimplementasikan interface:
  - **`public class`** `<NamaClass>` **`implements`** `<NamaInterface>`
- Nama interface sebaiknya dalam bentuk **adjective/kata sifat** jika merepresentasikan **kapabilitas**. Dapat juga menggunakan **kata benda**
- Contoh:

```
public interface Audible {
    void naikkanVolume();
    void turunkanVolume();
}
```

Implementasi Interface

- Menggunakan keyword **implements**
- Bila sebuah class mengimplementasikan suatu interface:
  - **Seluruh variable** dari interface akan dapat diakses oleh class tersebut
  - **Seluruh method** pada interface **harus diimplementasikan**
  - Bila class yang meng-implement interface **tidak mengimplementasikan semua method**, maka class tersebut harus dideklarasikan sebagai **abstract class**

```
public interface Audible {
    void naikkanVolume();
    void turunkanVolume();
}
```

```
public class TV extends AlatElektronik implements Audible {

    @Override
    public void naikkanVolume() {}

    @Override
    public void turunkanVolume() {}
}
```

Multiple Interface

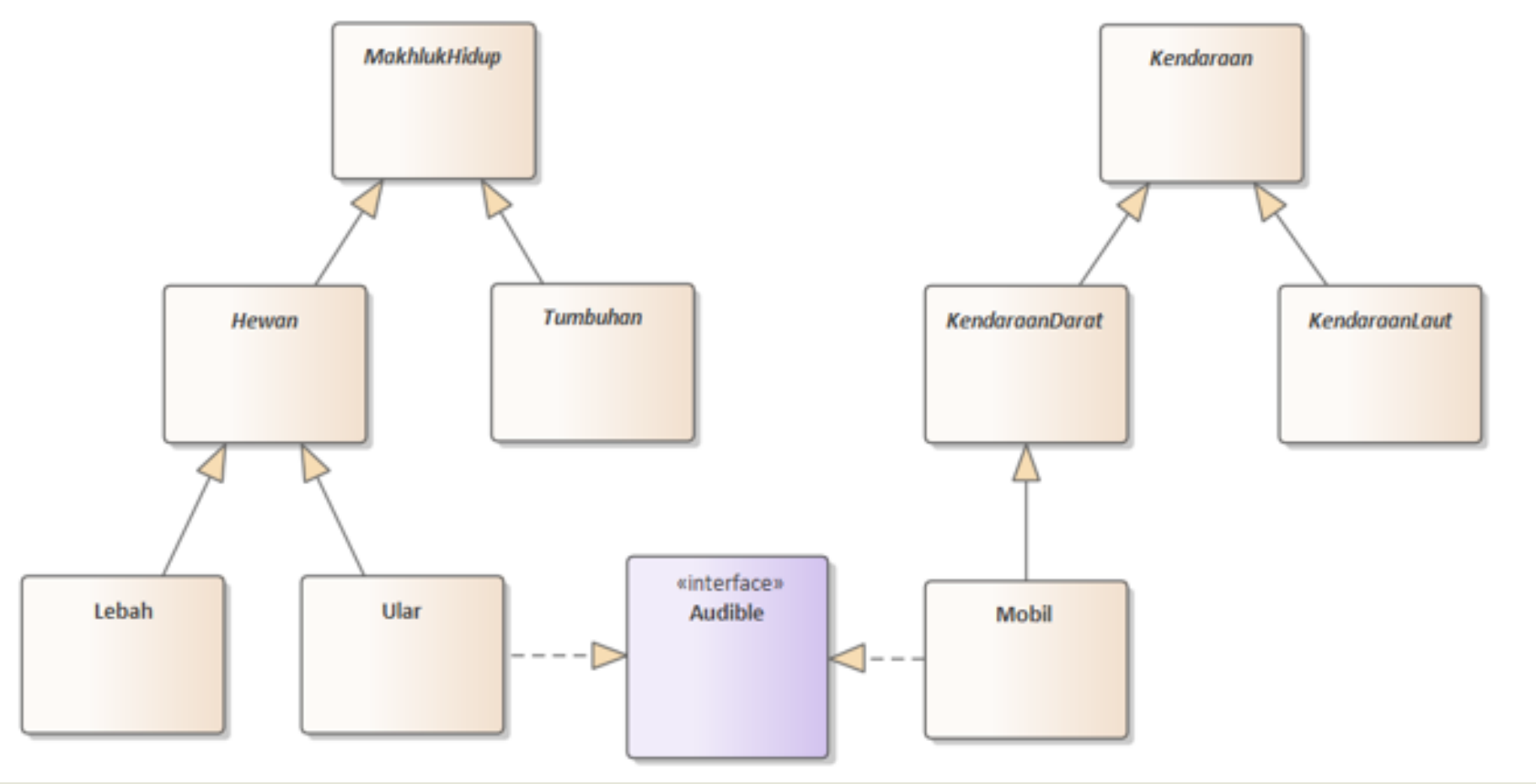
- Suatu class dapat meng-implement multiple interface
- Bila suatu class merupakan subclass dan meng-implement interface, maka keyword **extends** mendahului **implements**
- Contoh:

  public class PlainCandy extends GameItem implements Crushable, Movable

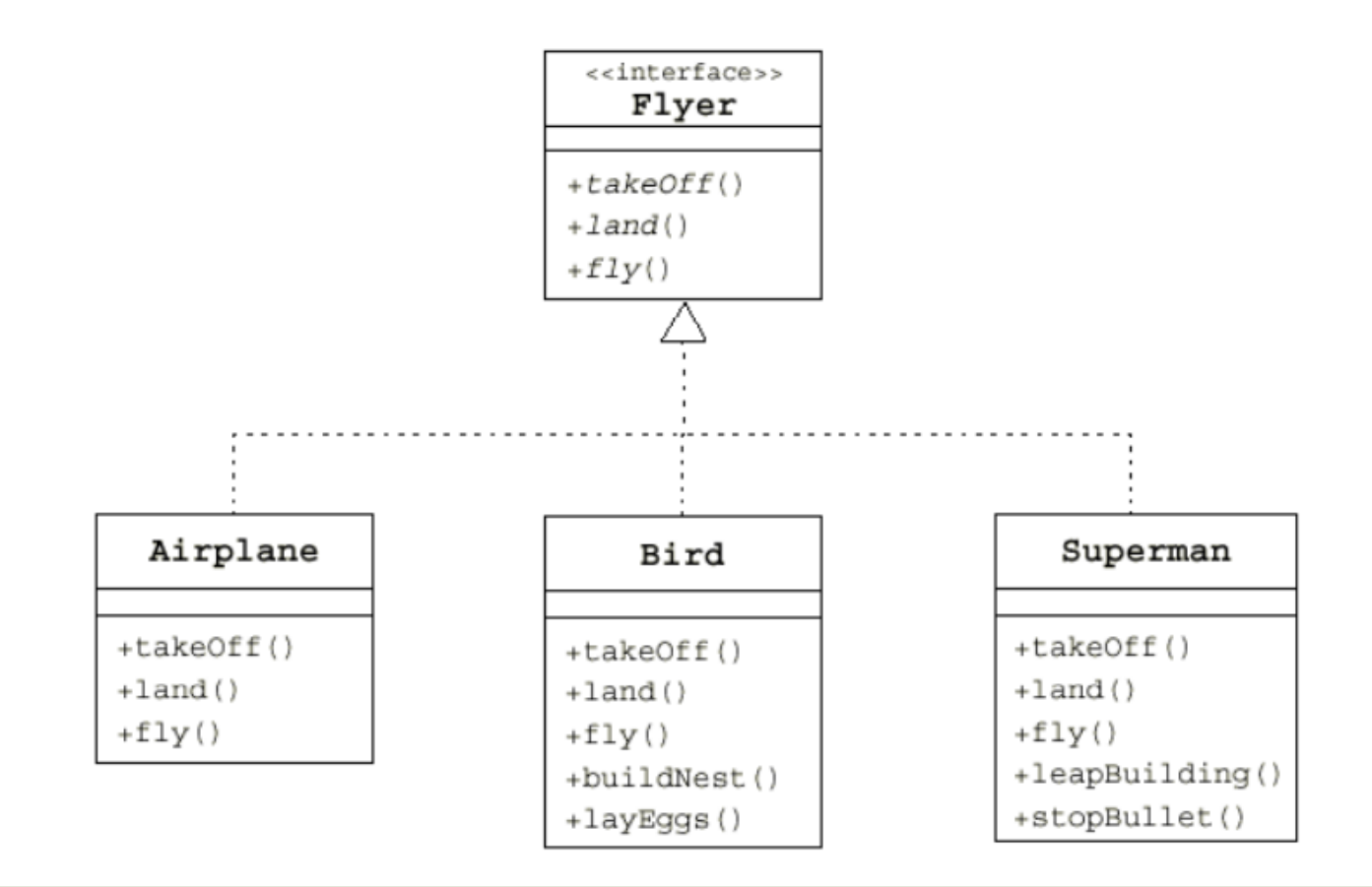
Abstract Class vs Interface

Abstract Class	Interface
Dapat memiliki **concrete** method atau **abstract** method	Hanya dapat memiliki **abstract** method
Level modifier atribut dan method: **public, protected, no-modifier, private**	Level modifier variable dan method hanya **public** (boleh tidak dituliskan)
Dapat memiliki **static/non-static, final/non final** variable	Hanya dapat memiliki **static dan final** variable
Method bisa **final/non final**	Method **tidak** boleh **final**
Digunakan untuk mendefinisikan hirarki class	Tidak mendefinisikan hirarki class/bukan bagian dari hirarki class

Interface tidak terikat pada hirarki



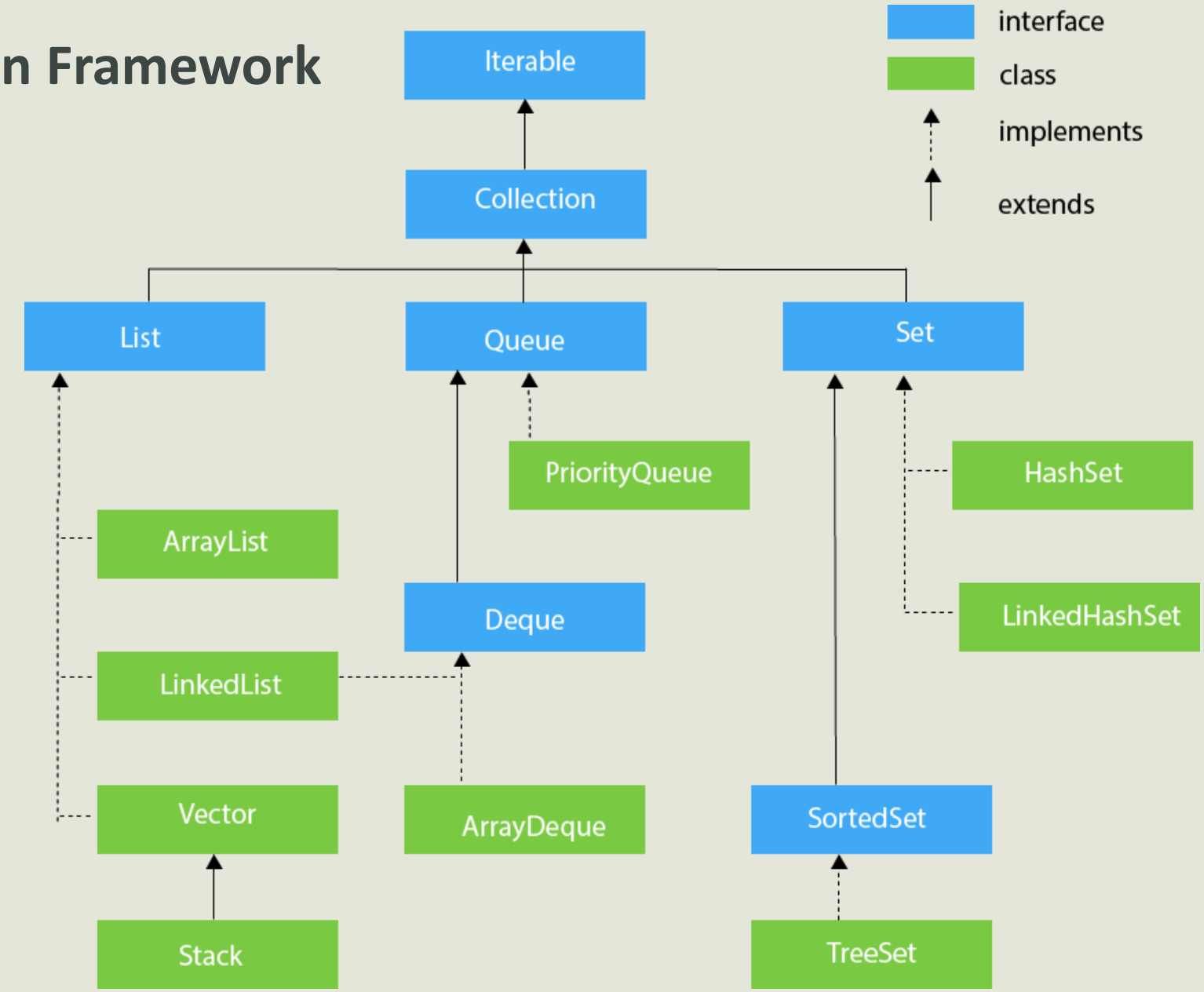
Interface tidak terikat pada hirarki

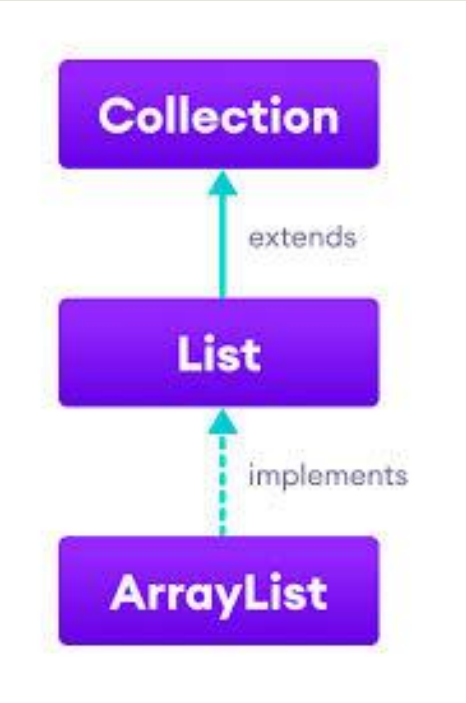


Extend untuk Interface

- Sebuah interface dapat meng-extend interface lain
- Class yang mengimplementasikan interface turunan **harus mengimplementasikan semua method** dari seluruh interface yang diwarisi.

Java Collection Framework





add(), remove(), clear(), contains(), isEmpty()

indexOf(), get()

add(), remove(), clear(), contains(), isEmpty(),
indexOf(), get()

Penggunaan Abstract Class vs Interface

- Common properties (and methods) → abstract class
- Common methods → interface

Java 8

- Sejak java 8 diperbolehkan adalanya default method
- Default method adalah **method di dalam interface** yang **memiliki implementasi langsung**, bukan hanya deklarasi.
- Interface tidak lagi hanya berisi *kontrak*, tapi bisa menyediakan **perilaku bawaan (default)**
- Sebelum Java 8, interface tidak bisa diubah tanpa memengaruhi semua class yang mengimplementasikannya.Kalau interface punya ratusan implementasi, menambah method baru akan merusak semua kode lama.
- Dengan default method, kita bisa **menambah method baru ke interface tanpa memaksa semua implementasi untuk menulis ulang method tersebut.**

Default Method

```
public interface Vehicle {

    String getBrand();
    String speedUp();
    String slowDown();

    default String turnAlarmOn() {
        return "Turning the vehicle alarm on.";
    }

    default String turnAlarmOff() {
        return "Turning the vehicle alarm off.";
    }
}
```

Kesimpulan

- Interface tidak dapat dibuat objek/*instance*-nya
- Interface merepresentasikan **kontrak/syarat** yang berisi sekumpulan *behavior* yang saling **terkait** untuk memenuhi suatu **kapabilitas**

Latihan

- Cari sebuah studi kasus dari interface kemudian gambarkan UML class diagramnya.