

Bahasa Pemrograman 2

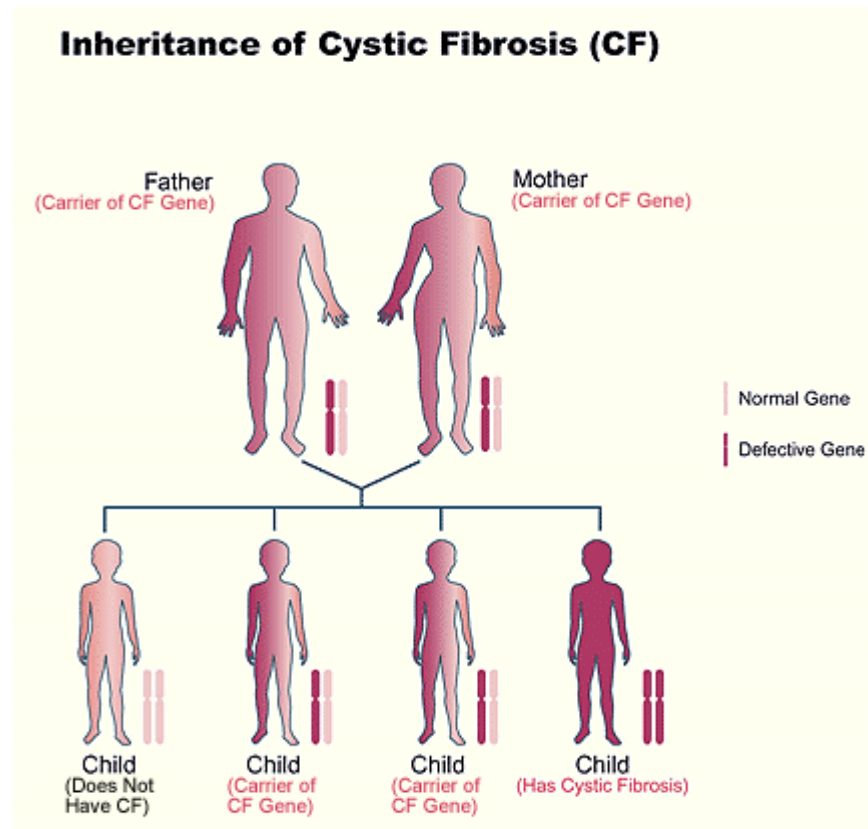
Inheritance

anton@ukdw.ac.id

Ciri khas OOP

- **Abstraksi** : Mendefinisikan obyek abstrak yang mampu melakukan kegiatan, mengubah state, dan berkomunikasi dengan obyek lain pada sistem
 - Membuat class yg terdiri dari atribut dan method
- **Enkapsulasi** : Menyembunyikan informasi dan detail implementasi sebuah method, serta mengatur akses terhadap atribut/method
 - Hak akses pada method
- **Polimorfisme** : Membuat obyek dari kelas dasar dapat berperilaku seperti obyek lain yang merupakan turunannya
 - Polimorfisme juga berarti banyak bentuk yg diimplementasikan pada multiple constructor class
- **Inheritance**: pewarisan atribut dan method dari class induk ke kelas anak

Inheritance



Relasi is-a

- Selain melakukan katagorisasi terhadap objek yang memiliki sekumpulan atribut dan perilaku yang sama, manusia sering melakukan pengelompokan terhadap objek yang memiliki kesamaan atas beberapa (**tidak semua**) atribut /perilaku
- Contoh : Pengelompokan atas kendaraan bermotor, kemudian meng-grupkannya berdasarkan suatu tipe atau jenis (mobil, truk, sepeda motor, dll.)
- Setiap subkatagori ini merupakan kelas atas objek-objek yang serupa.
 - Ada beberapa karakteristik yang di-share oleh semua kelompok.

Relasi is-a

- Relasi antar kelas-kelas ini disebut dengan **relasi “is-a”**
- Dalam setiap kasus, objek yang dikelompokkan bersama dalam satu sub-kategori merupakan anggota dari kategori yang lebih umum.
 - Mobil adalah (“is-a”) kendaraan bermotor
 - Truk adalah (“is-a”) kendaraan bermotor
 - Sepeda Motor adalah (“is-a”) kendaraan bermotor

Relasi is-a

- Objek yang dikelompokkan dalam satu kelas men-share sekumpulan atribut dan perilaku.
 - Jadi, seluruh objek kendaraan bermotor memiliki sekumpulan atribut dan perilaku yang juga dimiliki oleh /diturunkan kepada objek dari mobil.
- Keterkaitan antar kelas dalam relasi “is-a” berasal dari kenyataan bahwa **sub kelas memiliki atribut dan perilaku yang dimiliki oleh kelas induk, ditambah atribut dan perilaku yang dimiliki oleh sub kelas tersebut.**

Inheritance

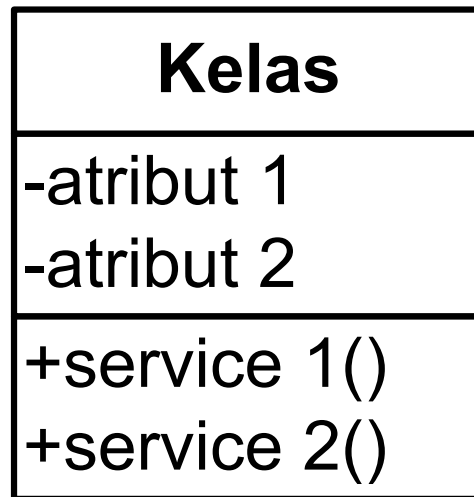
- Superclass (“kelas dasar” atau “kelas induk”)
 - Merupakan kelas yang lebih general dalam relasi “is-a”
- Subclass (“kelas turunan” atau “kelas anak”)
 - Merupakan kelas yang lebih spesifik dalam relasi “is-a”
 - Objek yang dikelompokkan dalam sub kelas memiliki atribut dan perilaku kelas induk, dan juga atribut dan perilaku tambahan. (**Jadi, kumpulan atribut dan perilaku sub kelas lebih besar dari super kelas-nya**)

Inheritance

- Relasi “is-a” antar superclass dan subclasses-nya disebut dengan **pewarisan** atau **inheritance**
- Kita mengatakan subclass “mewarisi” suatu superclass (atau juga bisa dikatakan sebuah subclass “turunan dari” suatu superclass)

Class Diagram [1]

- Contoh Class Diagram:



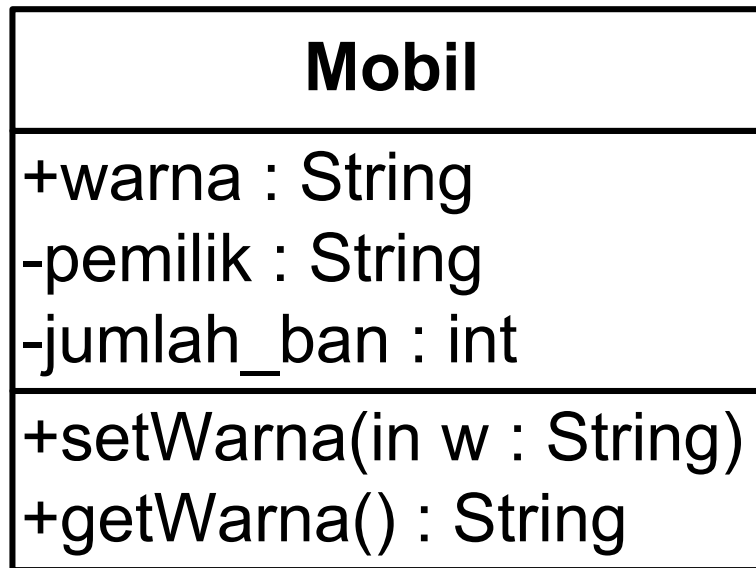
Class Diagram [2]

- Sebagai contoh, kita memiliki program seperti berikut:

```
class Mobil {  
    public String warna;  
    private String pemilik;  
    private int jumlah_ban;  
  
    public void setWarna(String w) {  
        this.warna = w;  
    }  
    public String getWarna() {  
        return this.warna;  
    }  
}
```

Class Diagram [3]

- maka bisa kita buat class diagramnya sebagai berikut:



- Perhatikan penulisan atribut dan service/methods-nya!

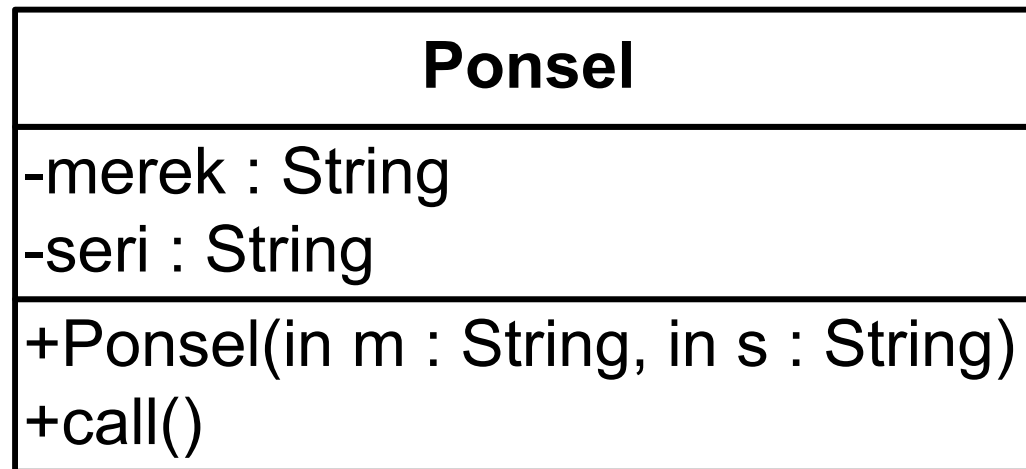
Class Diagram [4]

- Bagaimana dengan yang berikut?

```
class Ponsel {  
    private String merek;  
    private String seri;  
  
    public Ponsel(String m, String s) {  
  
    }  
  
    public void call() {  
        System.out.println("Calling...");  
    }  
}
```

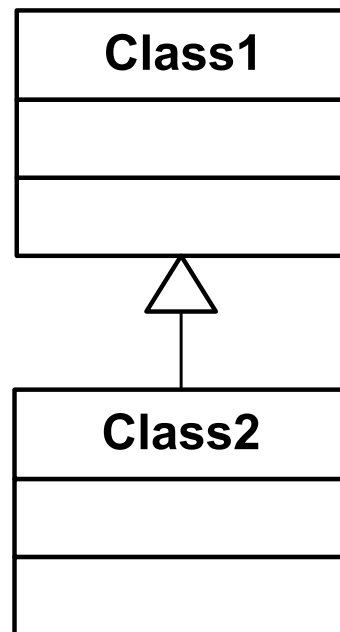
Class Diagram [5]

- Class Diagramnya dapat digambarkan sebagai berikut:



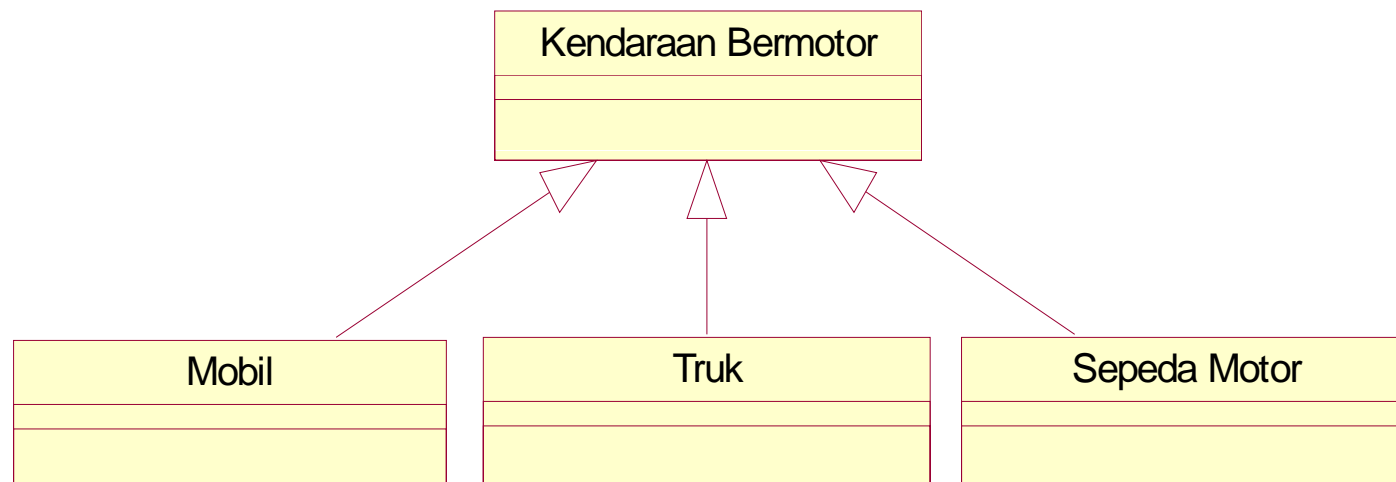
Class Diagram [6]

- Bagaimana penggambaran class diagram untuk inheritance?
- Anggap ada 2 buah class yang salah satunya adalah superclass dari yang lainnya:



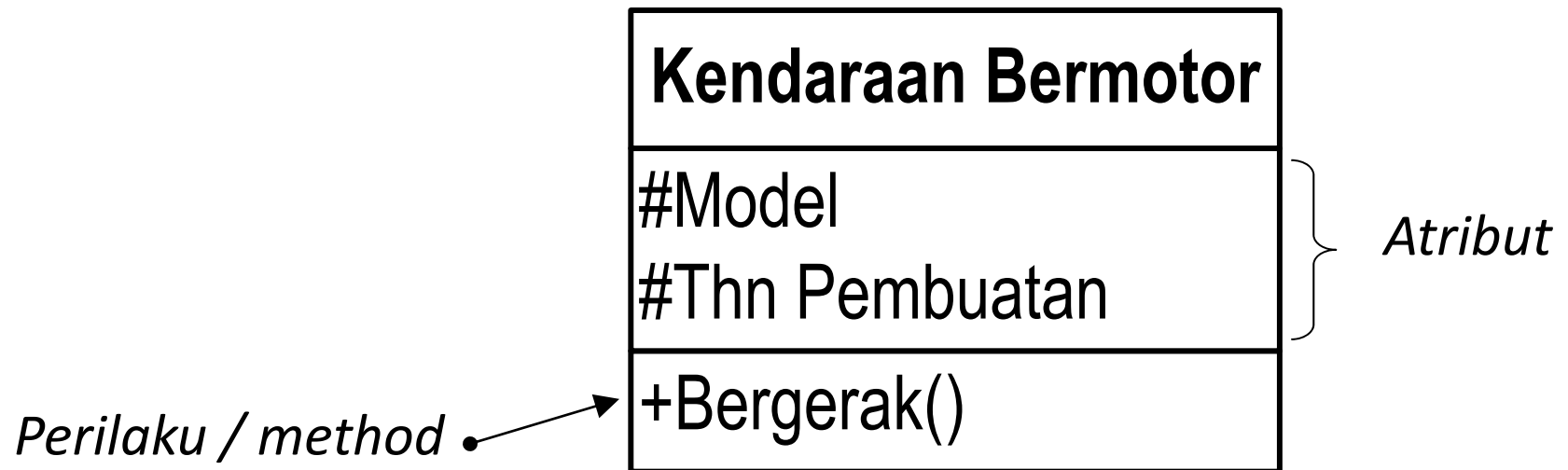
Object Relationship Diagrams

- Super kelas memiliki beberapa sub kelas



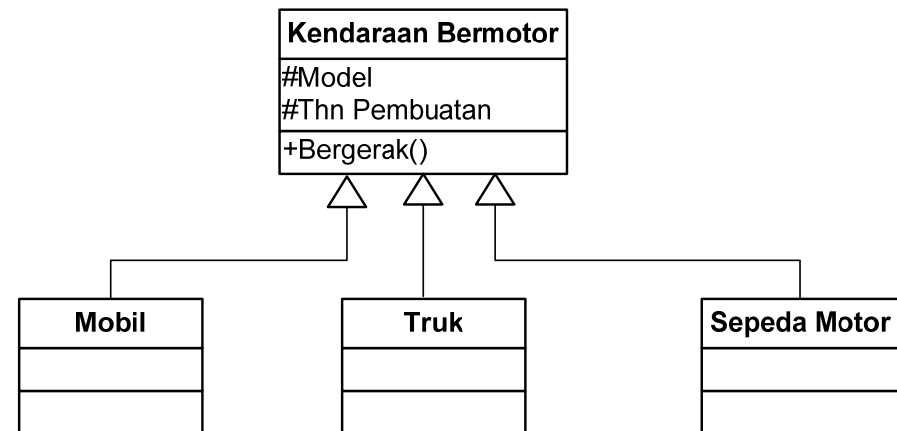
Object Relationship Diagrams

- Setiap objek memiliki atribut dan perilaku



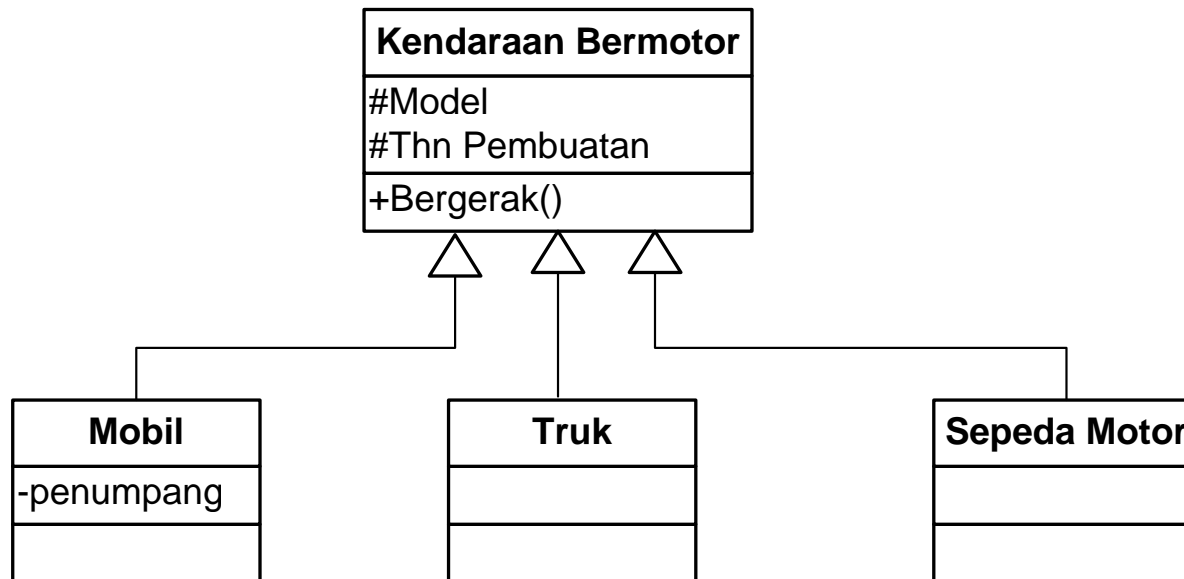
Object Relationship Diagrams

- Mobil, truk dan sepeda motor memiliki atribut *model* dan *Thn Pembuatan* serta perilaku *bergerak*
- Setiap sub kelas harus berbeda satu dengan yang lainnya. Jika tidak, tidak ada alasan untuk membuat sub kelas tersebut
- Misalkan semua mobil memiliki atribut tambahan *penumpang* (yang menunjukkan jumlah maks yang bisa diangkut)



Object Relationship Diagrams

- Maka gambar menjadi

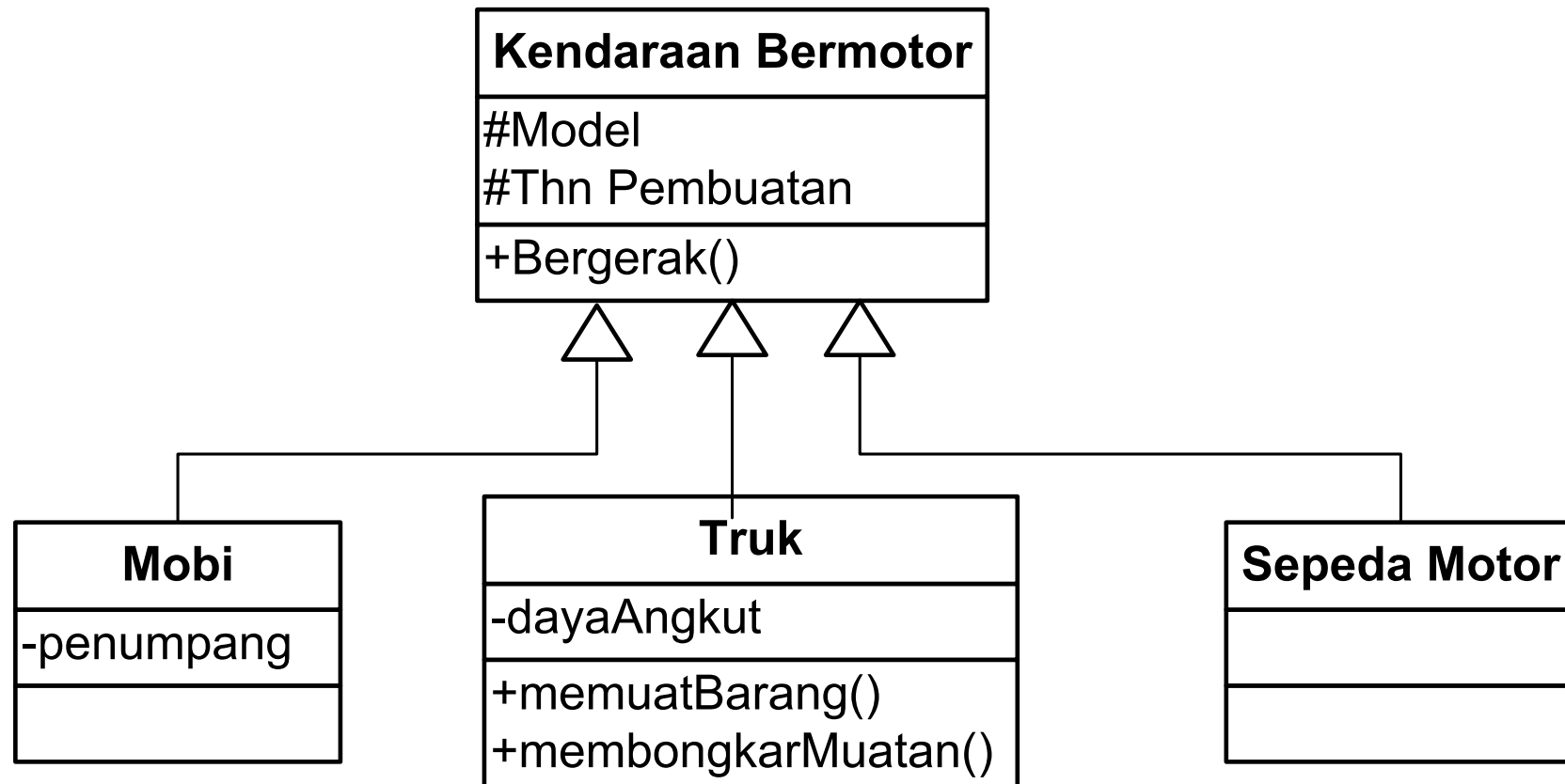


Object Relationship Diagrams

- Misalkan truk memiliki atribut tambahan daya angkut dan perilaku memuat barang dan membongkar muatan, Maka diagram lengkap akan menunjukkan:
 - Semua truk akan memiliki atribut model, tahun pembuatan dan daya angkut, dan perilaku bergerak, memuat barang, dan membongkar muatan

Object Relationship Diagrams

- *Gambar*

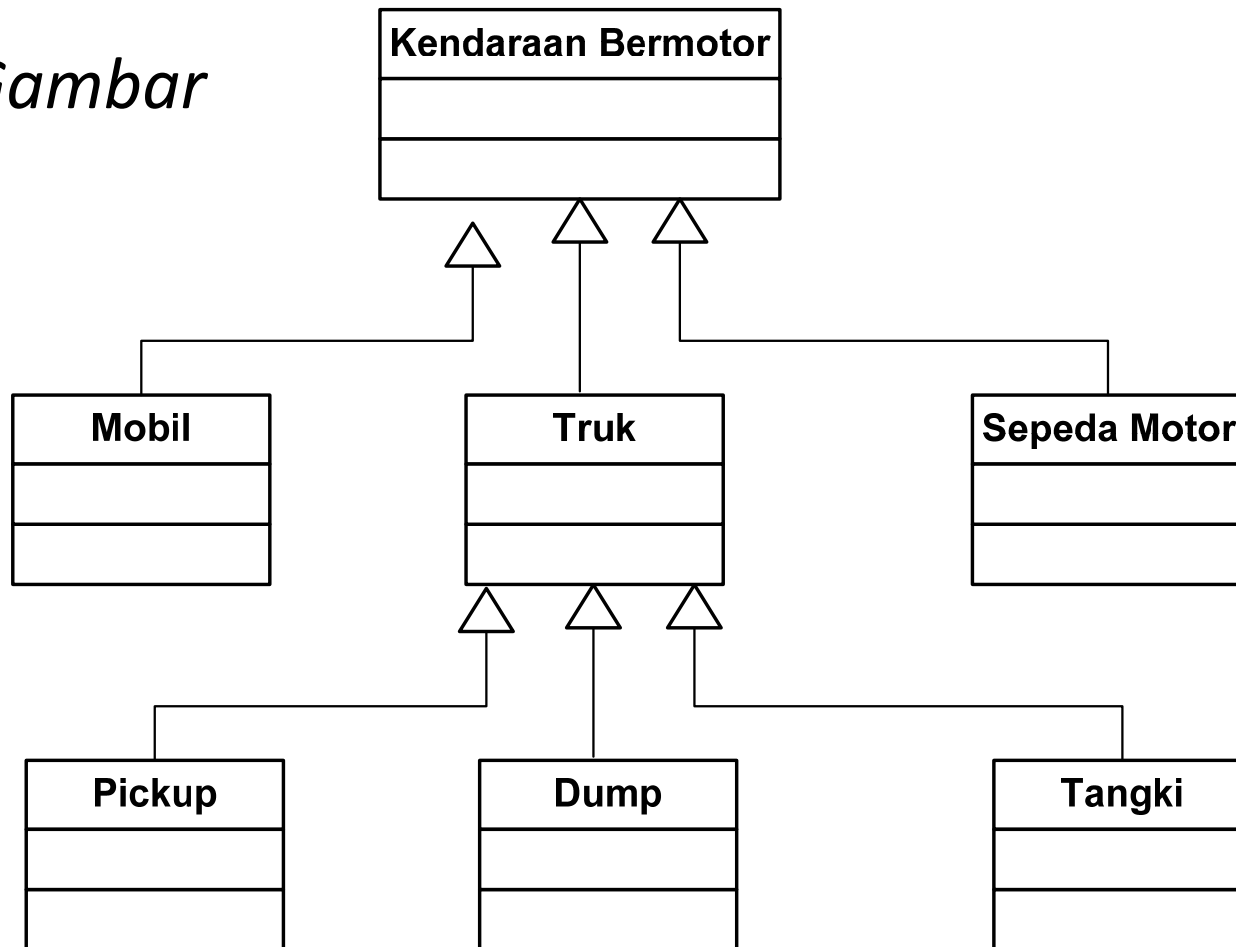


Hirarki Pewarisan

- Relasi pewarisan selalu ditunjukkan dengan sub kelas diletakkan di bawah super kelas, untuk menekankan sifat hirarkis relasi
- Kita dapat memiliki multi layer pewarisan
 - Misal : kelas truk bisa saja menjadi super kelas dari truk pickup, truk dump , dan truk tangki

Hirarki Pewarisan

- *Gambar*



Hirarki Pewarisan

- Dalam hirarki pewarisan, perlu diketahui bahwa pada setiap level pewarisan, sub kelas melanjutkan pewarisan dari super kelas-nya.
 - Truk Pickup adalah sebuah truk dan juga kendaraan bermotor
- Berarti setiap sub kelas akan mendapatkan atribut dan perilaku dari setiap super kelas yang berada di level atasnya
 - Truk pickup akan memiliki seluruh atribut dan perilaku suatu kendaraan bermotor, ditambah dengan atribut dan/atau perilaku suatu truk, ditambah dengan atribut dan perilaku dari truk pickup itu sendiri

Inheritance

- Merupakan merupakan pewarisan atribut-atribut dan dan method-method dari dari sebuah sebuah class ke class lainnya.
- Class yang memberi warisan => **superclass**
- Class yang diberi warisan => **subclass**
- Contoh:
 - Superclass => sepeda
 - Subclass => sepeda gunung, sepeda balap, sepeda motor
 - Keyword pada Java = **extends**

Inheritance

- Keuntungan:
 - Memberikan ciri khas pada masing-masing subclass
 - Superclass mewariskan atribut dan methodnya ke subclass sehingga menerapkan **reuse**
- Pada inheritance juga dikenal adanya **overriding**
 - Method yang sama nama dan tipenya tapi di kelas berbeda namun masih dalam satu hubungan keturunan
 - Jika ada method di kelas parent yang sudah didefinisikan, dan didefinisikan ulang, maka method pada kelas anak akan *menutup* method parent, kecuali dibuat **final**

Implementasi dalam Bahasa Pemrograman (Java)

Sintaks

```
class Anak extends Induk {  
  // deklarasi badan kelas  
  
}
```

Implementasi dalam Bahasa Pemrograman (Java)

```
class KendaraanBermotor {  
    protected String model;  
    protected int thnPembuatan;  
  
    public KendaraanBermotor(String m,  
int thn) {  
        model = m;  
        thnPembuatan = thn;  
    }  
    public void bergerak() {  
        System.out.println ("Kendaraan  
Motor : bergerak \n");  
    }  
}
```

```
class Mobil extends KendaraanBermotor {  
    private int penumpang;  
  
    public Mobil(String m, int thn, int  
pnmp)  
    {  
        super (m)  
        penumpang =pnpm;  
    }  
    public void memuatPenumpang() {  
        System.out.println ( "Memuat  
Penumpang \n");  
    }  
}
```

Aksesabilitas Anggota (1)

- private
 - Tidak dapat diakses oleh kelas lain
 - Akses/manipulasi melalui fungsi anggota
- protected
 - Dapat diakses oleh kelas turunan
- public
 - Dapat diakses oleh sebarang kelas

Aksesabilitas Anggota (2)

Catatan

- Kelas turunan dapat mengakses atribut kelas induk dengan yang protected atau public.
- Atribut yang private tidak diturunkan dari kelas induk ke kelas anak.

Aksesabilitas Anggota (3)

Catatan

- Kelas turunan DAPAT mengakses setiap public member kelas dasar, kelas lain juga DAPAT mengakses member kelas dasar secara langsung.
- Kelas turunan TIDAK DAPAT mengakses private member kelas dasar, kelas lain juga TIDAK DAPAT mengakses member kelas dasar secara langsung.
- Kelas turunan DAPAT mengakses setiap protected member kelas dasar, TETAPI kelas lain TIDAK DAPAT mengakses member kelas dasar secara langsung.

Manfaat Inheritance

- Tanpa inheritance, maka semua attribute dan method yang pernah dibuat dan butuhkan kelas lain, **harus ditulis ulang** seluruhnya.
- Dengan inheritance, seorang programmer ingin memodifikasi suatu attribute atau method yang dimanfaatkan subclass, maka dilakukan modifikasi attribute dan method tersebut pada class supernya.

Super

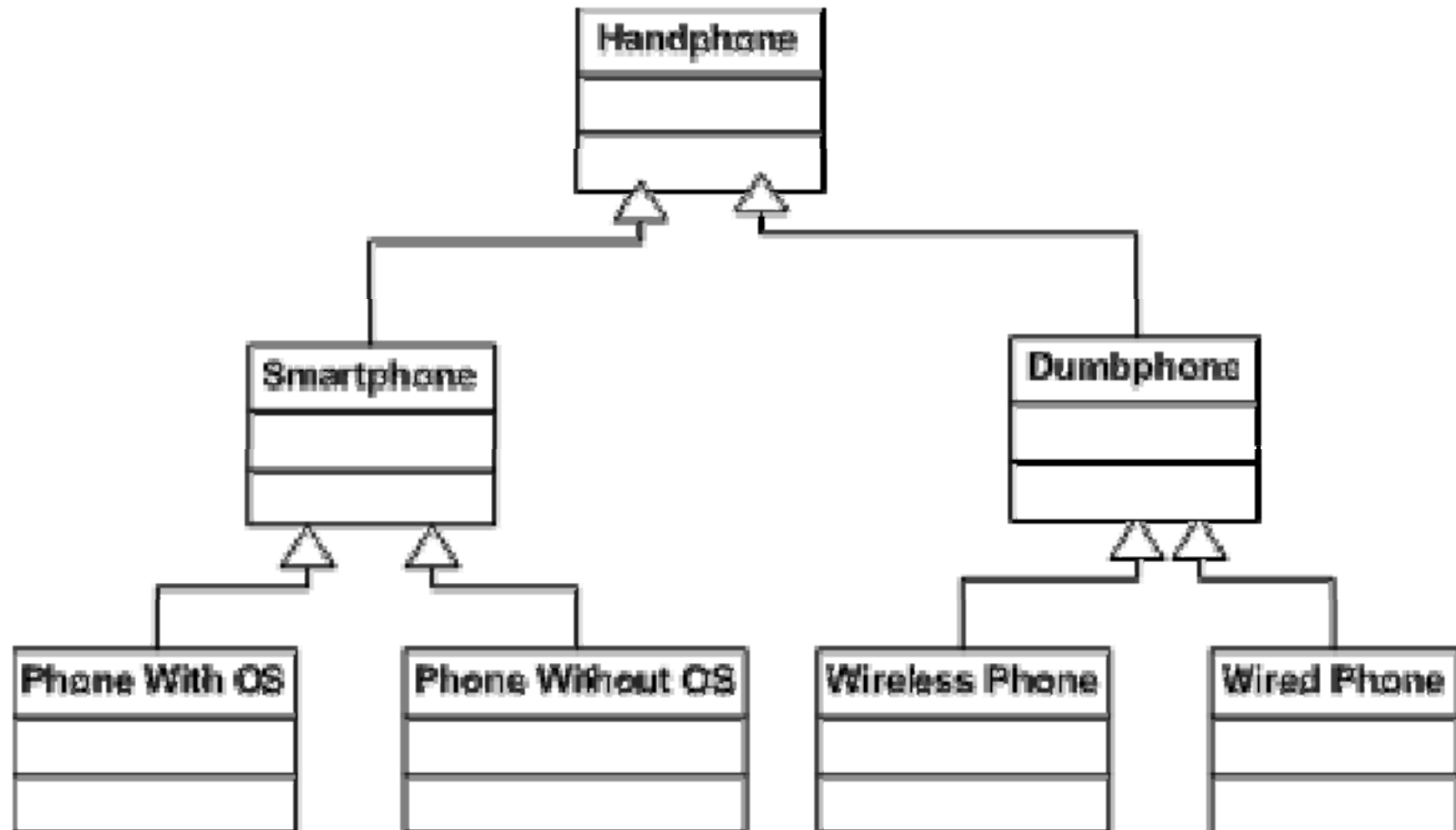
- Super dapat digunakan untuk memanggil konstruktor yang ada pada **superclass**.
- Penggunaan super **harus dilakukan saat awal di dalam sebuah konstruktor** sebuah kelas

```
class <nama_kelas> extends <nama_superclass>
{
    //konstruktor <nama_kelas>
    public <nama_kelas>
    {
        super(<parameter_list>)
    }
}
```

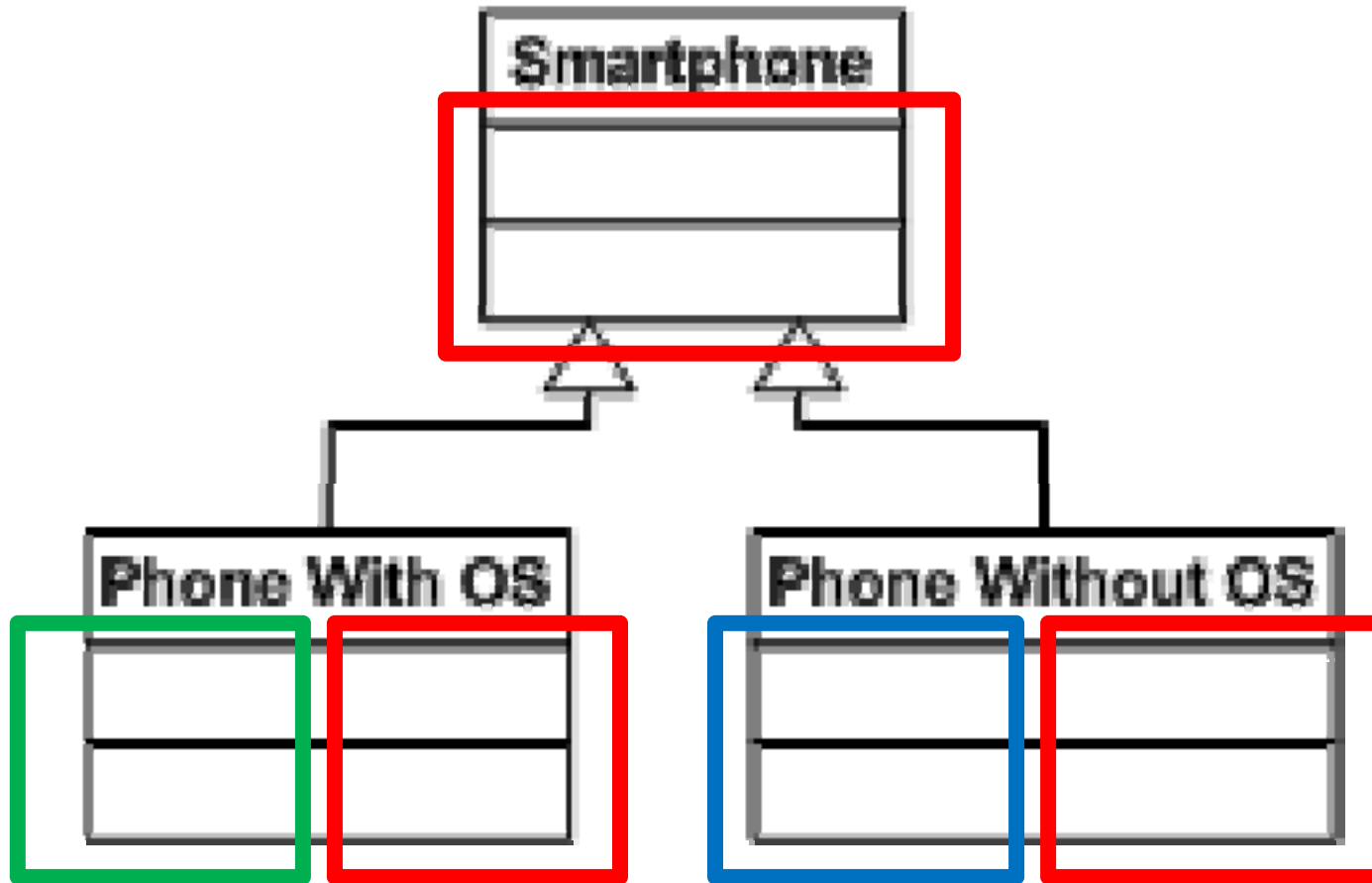
Contoh Inheritance

```
class CivitasAkademika {
    CivitasAkademika() {
        System.out.println("Semua warga universitas");
    }
}
class Staff extends CivitasAkademika {
    Staff() {
        System.out.println("yang mencari sesuap nasi");
    }
}
public class StaffEdukatif extends Staff {
    StaffEdukatif() {
        System.out.println("sebagai pengajar");
    }
    public static void main(String[] args) {
        StaffEdukatif x = new StaffEdukatif();
    }
}
```

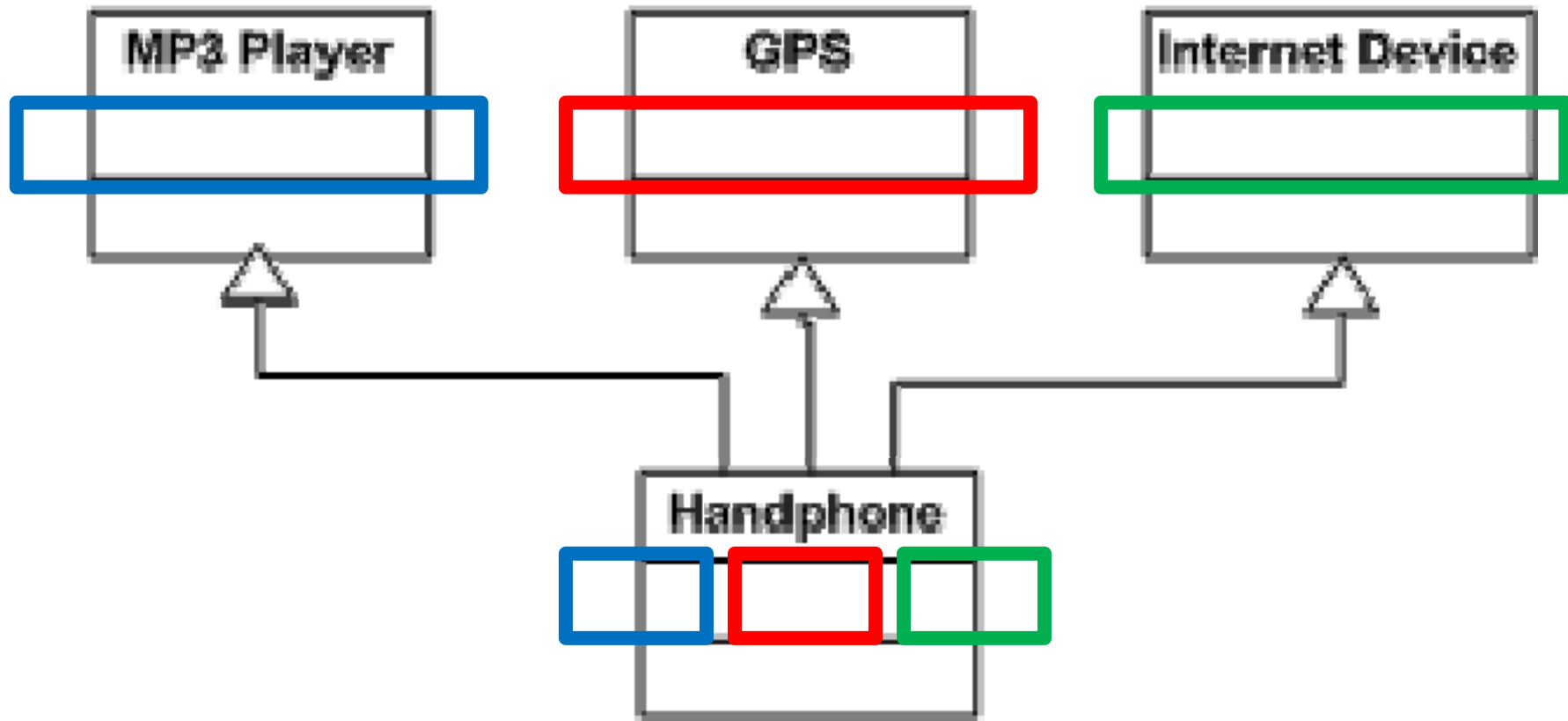
Inheritance



Inheritance



Multiple Inheritance

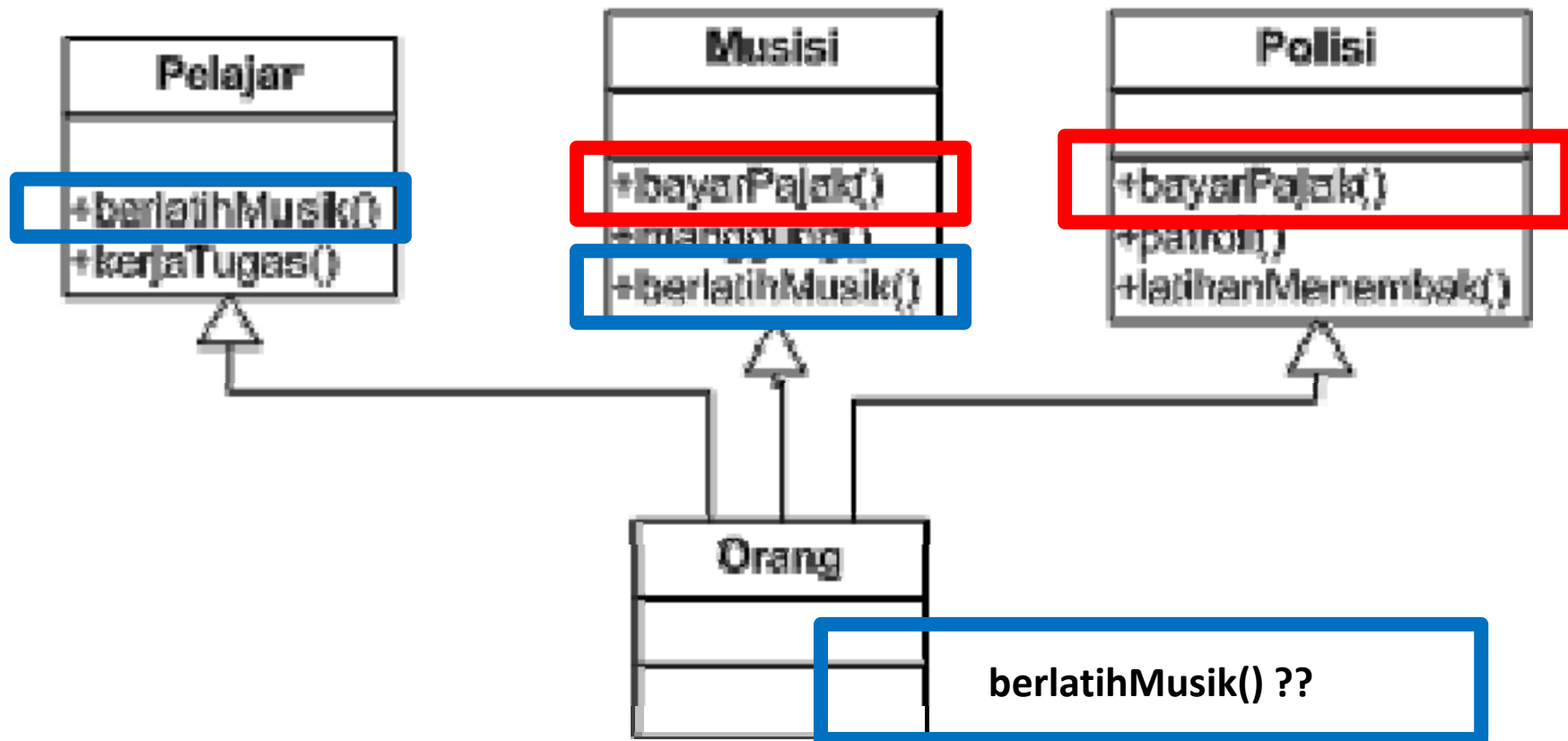


Multiple Inheritance

- **Java tidak mendukung Multiple Inheritance**
- C++ mendukung Multiple Inheritance
- Multiple Inheritance bisa menimbulkan ambiguitas (***diamond problem***)

Multiple Inheritance

- Misalkan ada 3 Class seperti berikut:



Interface

- Java hanya mendukung single inheritance, tidak mendukung multiple inheritance
- Java menggunakan interface untuk melakukan multiple inheritance → **MULTIPLE INTERFACE INHERITANCE**

NEXT

- Interface & Abstract