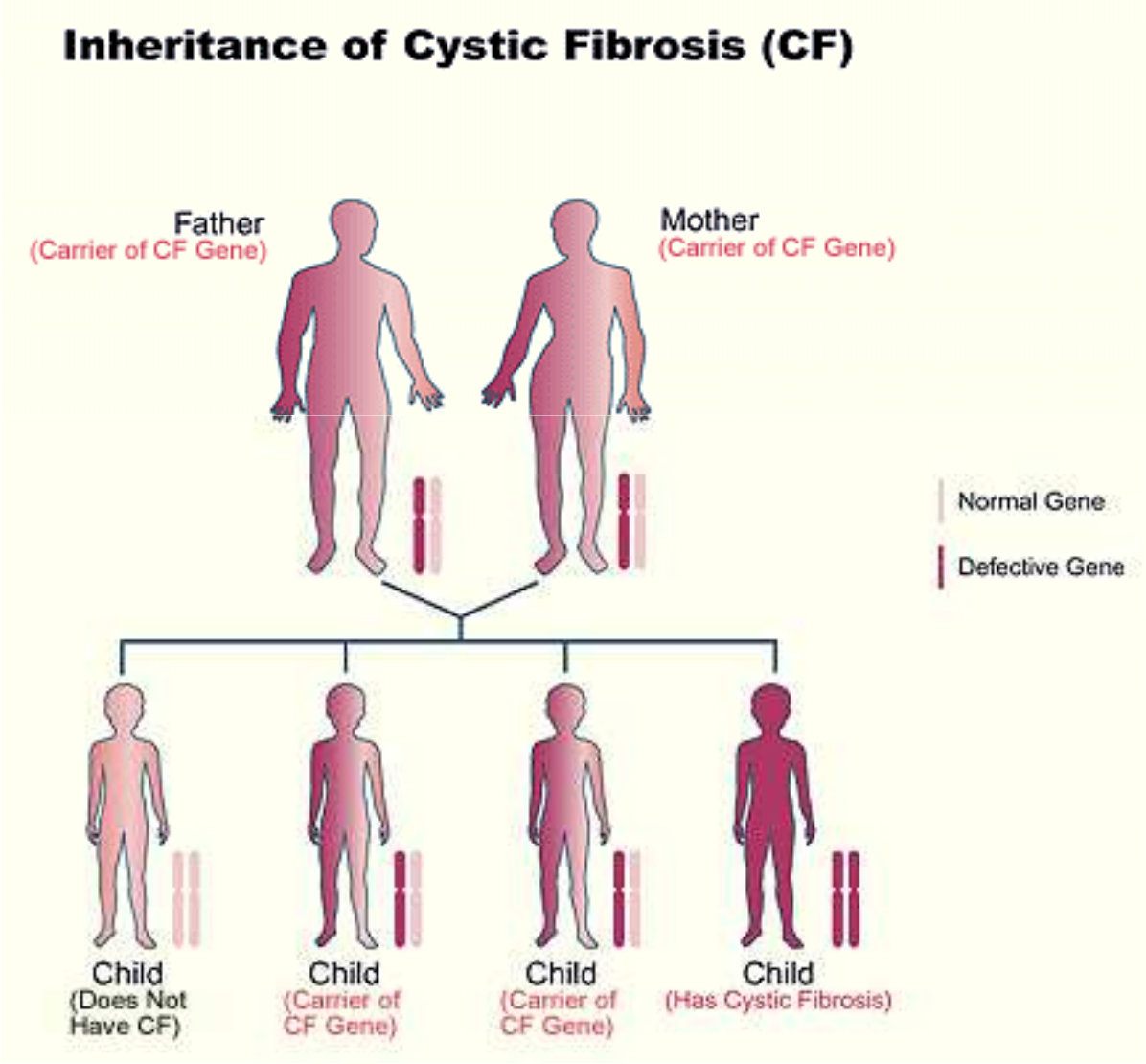
Pemrograman Berorientasi Obyek

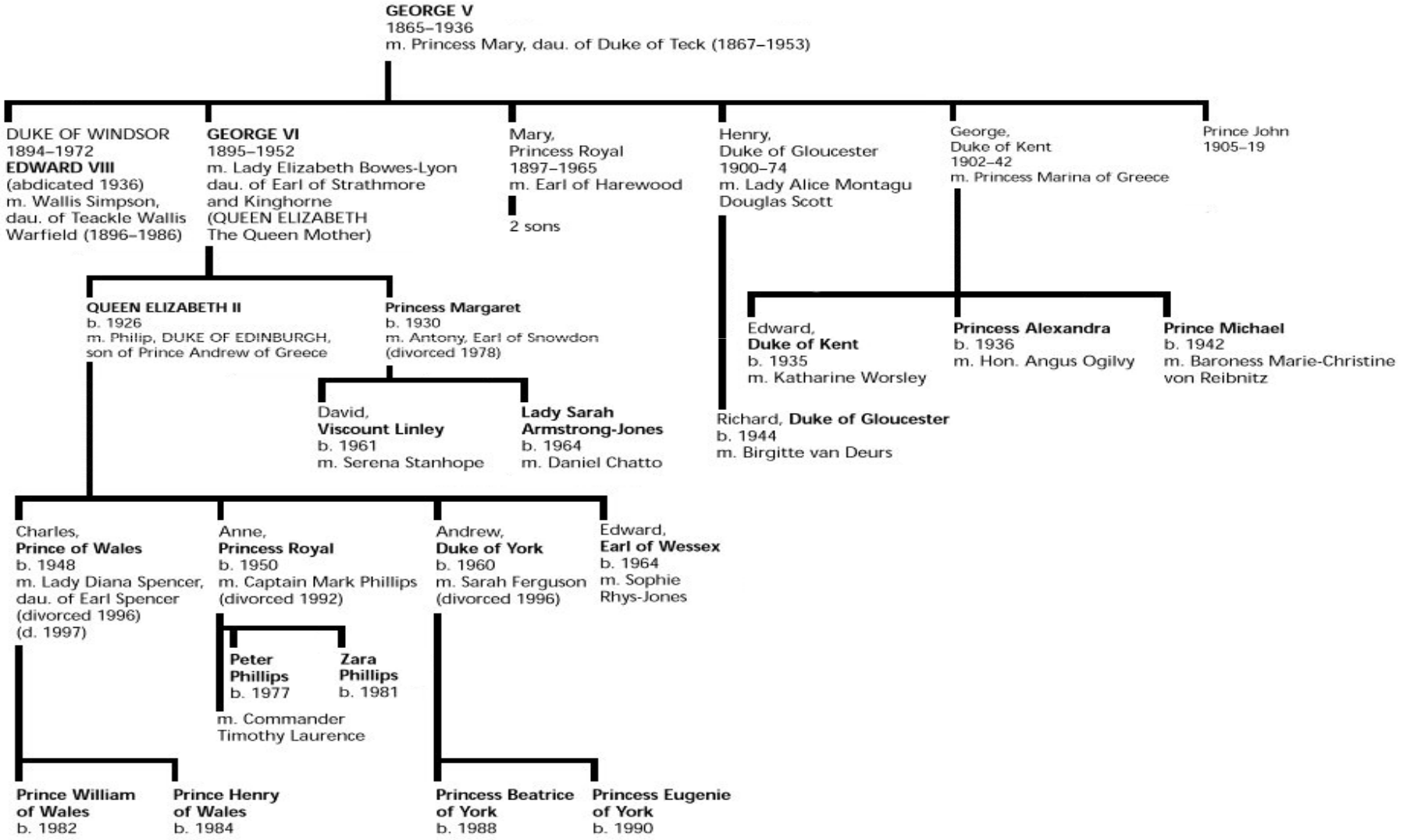
Inheritance

anton@ukdw.ac.id

Inheritance



Silsilah Pohon Keluarga



Relasi **is-a**

- Selain melakukan kategorisasi terhadap objek yang memiliki sekumpulan atribut dan perilaku yang sama, manusia sering melakukan pengelompokan terhadap objek yang memiliki kesamaan atas beberapa (**tidak semua**) **atribut dan perilaku**

- Contoh : Pengelompokan atas kendaraan bermotor, kemudian meng-grupkannya berdasarkan suatu tipe atau jenis (mobil, truk, sepeda motor, dll)

- Setiap subkategori ini merupakan class 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

- Take an **existing** object type (collection of fields and methods) and **extend** it.
  - create **a special version** of the code **without re-writing** any of the existing code (or even explicitly calling it!)
  - End result is a **more *specific* object type**, called the **sub-class** / derived class / child class.
  - The original code is called the **super class** / parent class / base class.

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 luas dari super kelas-nya**

Inheritance

- Merupakan merupakan pewarisan 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**

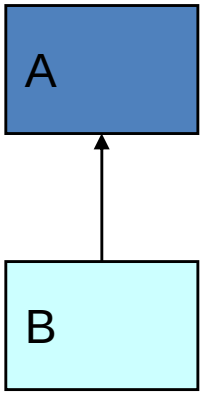
Inheritance

- 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 ***menimpa*** method parent, kecuali dibuat **final**

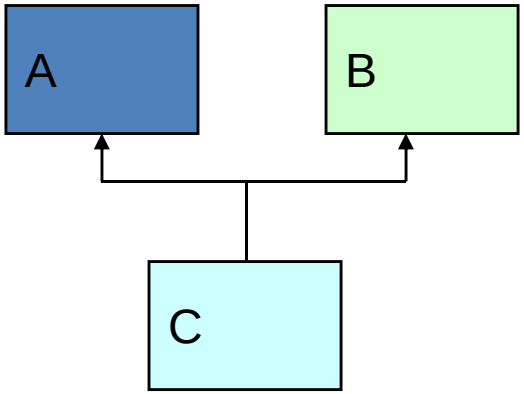
Bentuk-bentuk Inheritance

- The different forms of inheritance are:
  - **Single inheritance (only one super class)**
  - **Multiple inheritance (several super classes)**
  - Hierarchical inheritance (one super class, many sub classes)
  - Multi-Level inheritance (derived from a derived class)
  - Hybrid inheritance (more than two types)
  - Multi-path inheritance (inheritance of some properties from two sources)

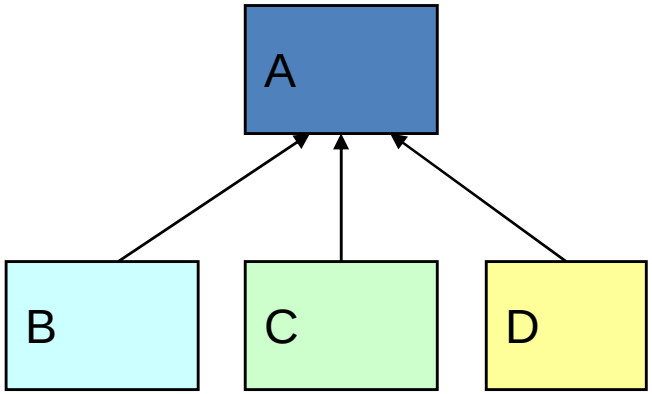
Bentuk-bentuk Inheritance



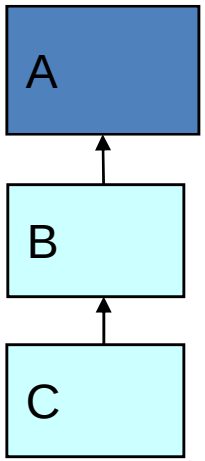
(a) Single Inheritance



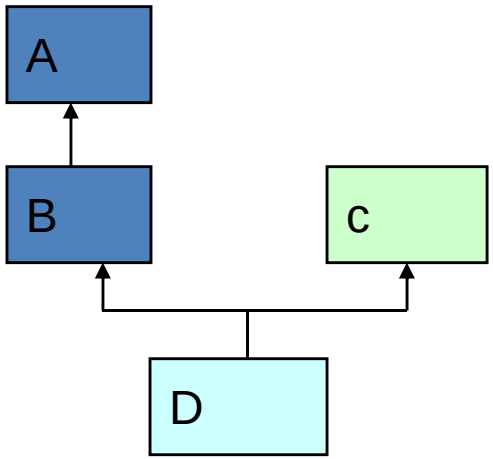
(b) Multiple Inheritance



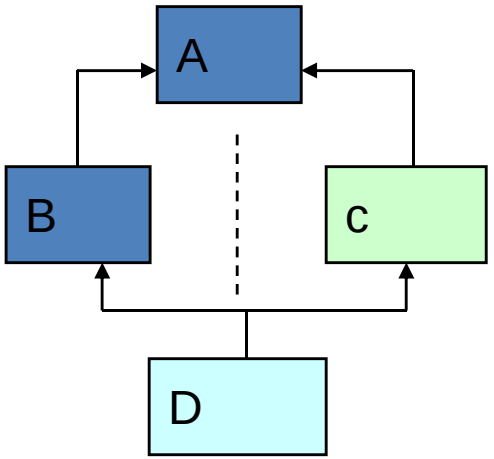
(c) Hierarchical Inheritance



(a) Multi-Level Inheritance



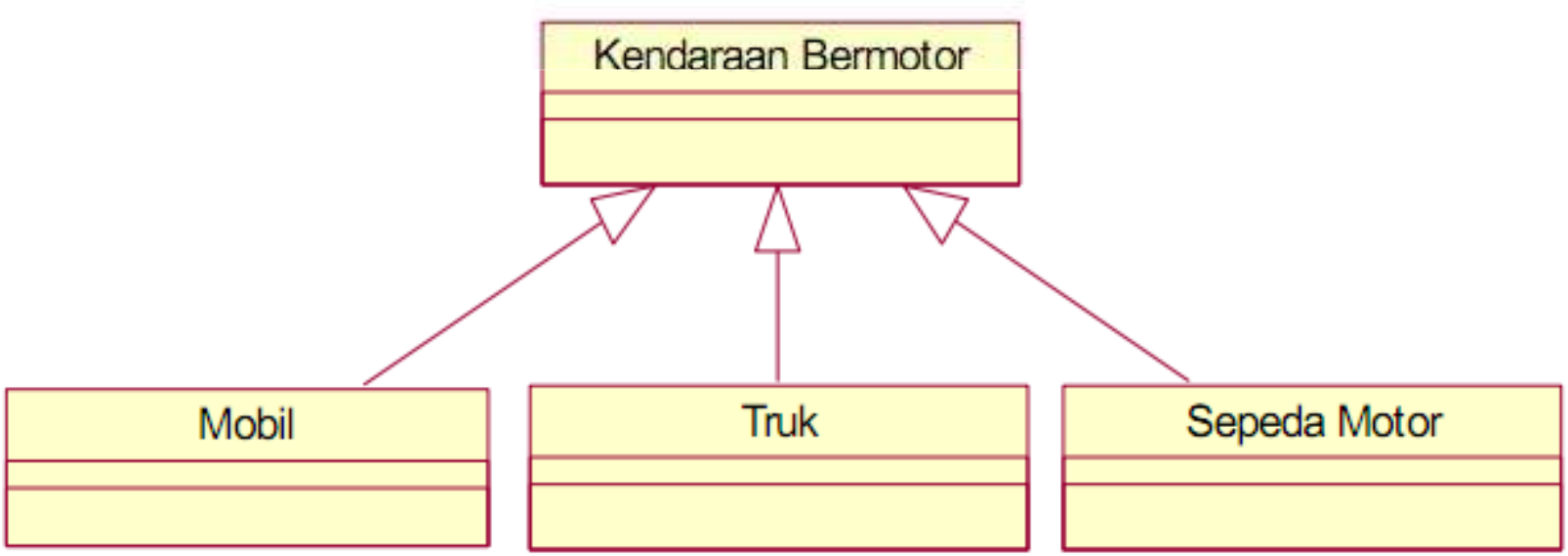
(b) Hybrid Inheritance



(b) Multipath Inheritance

UML

Pengelompokan juga bisa menggunakan dasar kesamaan
Menggunakan dasar generalisasi dan spesialisasi

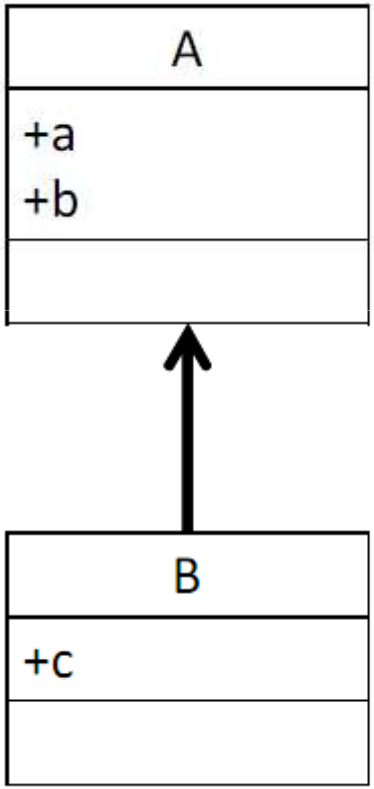


Implementasi dalam Bahasa Pemrograman (Java)

Sintaks

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

}
```

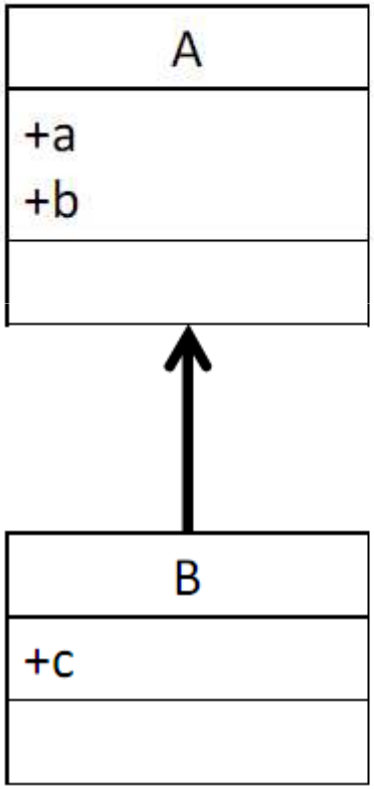
Implementasi




```
public class A {
        public int a;
        public int b;
}
```

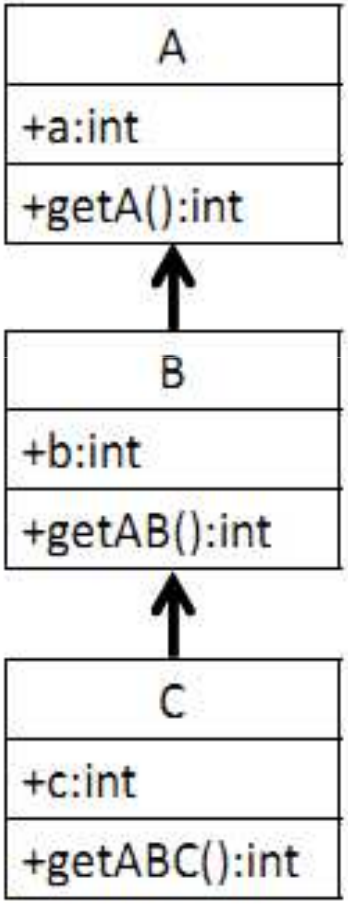
```
public class B extends A {
    public int c;
}
```

Tester




```
public class Tester {
    public static void main(String[] args) {
        B test = new B();
        System.out.println(test.a);
        System.out.println(test.b);
        System.out.println(test.c);
    }
}
```

Inheritance berurutan

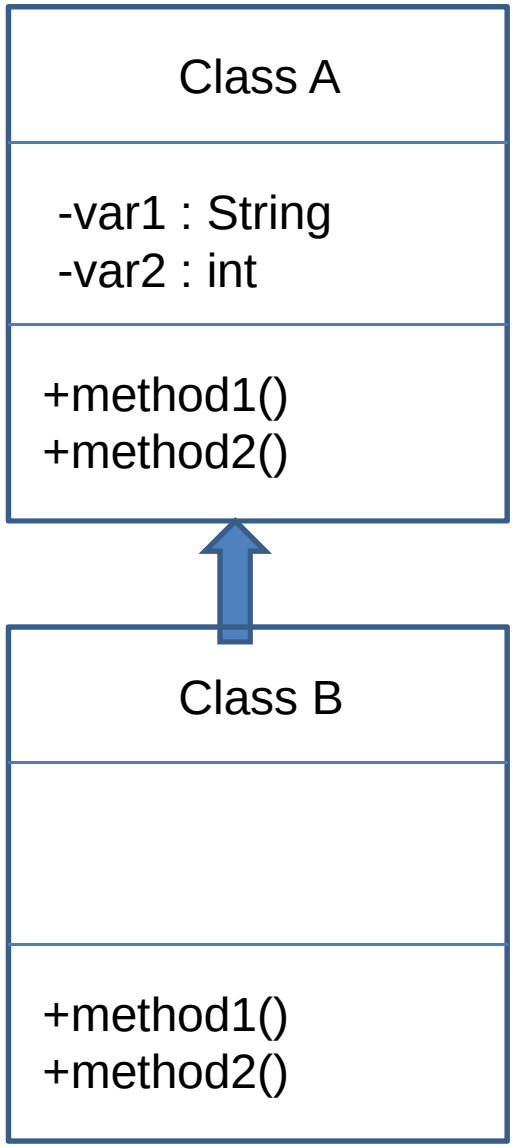



```
public int getA() {
    return a;
}

public int getAB() {
    return (a + b);
}

public int getAB() {
    return (a + b + c);
}
```

Review: Aksesabilitas Anggota

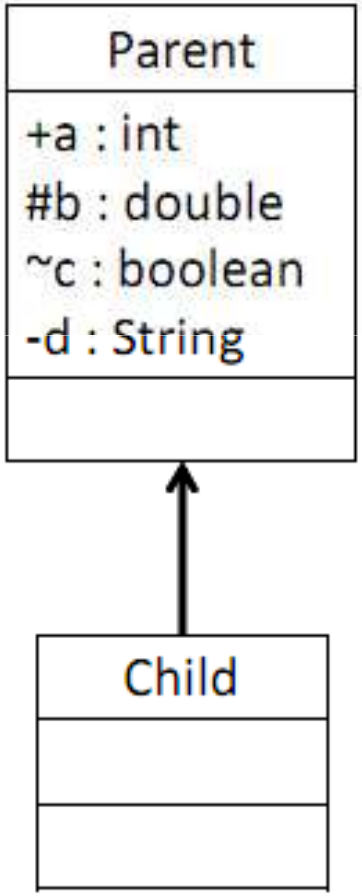
- private
  - Tidak dapat diakses oleh kelas lain
  - Akses/manipulasi melalui fungsi anggota
- protected
  - Dapat diakses oleh **kelas turunan**
- public
  - Dapat diakses oleh sembarang kelas
- Kelas turunan dapat mengakses atribut kelas induk dengan yang **protected** atau **public**.
- Atribut yang **private tidak** diturunkan dari kelas induk ke kelas anak.

UML



Class B pasti mendapat ‘warisan’ method **public**:
-Method1
-Method2
Sedangkan atribut **private** tidak akan diwariskan

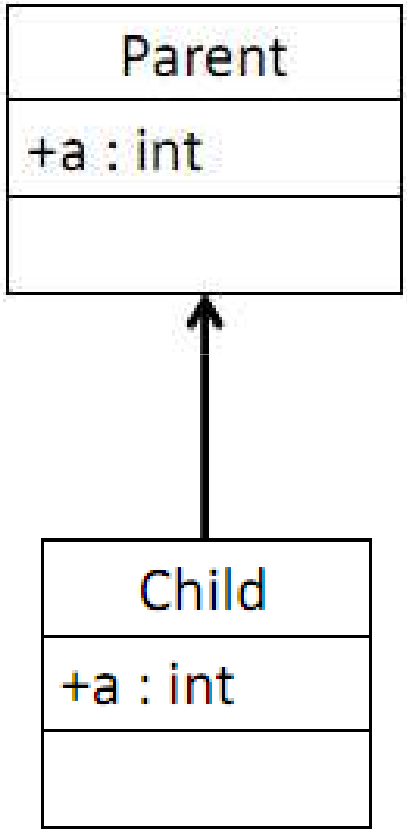
Pewarisan variabel class



Yang diwariskan adalah a,b dan c

Field d tidak diwariskan karena ditandai Sebagai private

Pewarisan variabel class

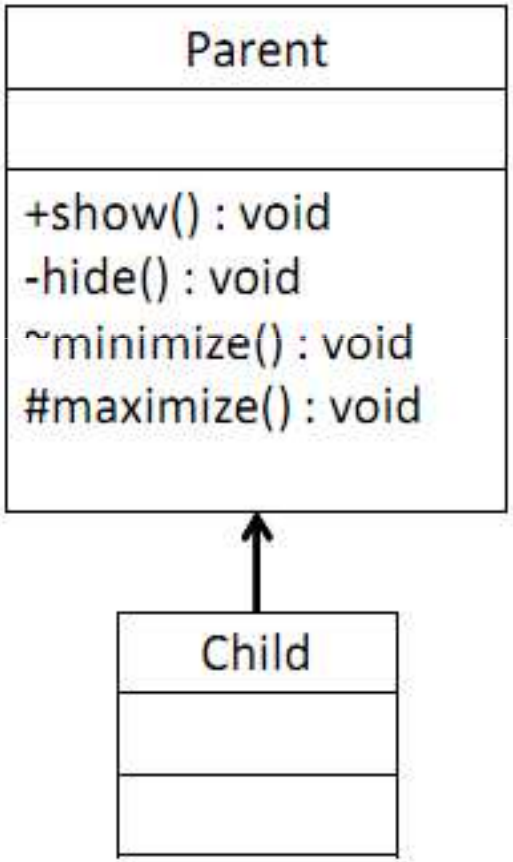


Field a dari class Parent diwariskan ke class Child

Pada class Child sekarang ada dua field bernama a

Hal seperti ini sebaiknya dihindari !

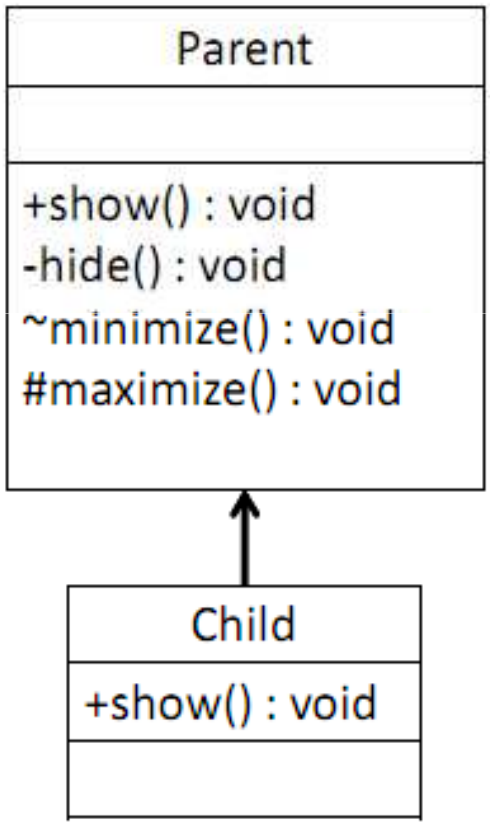
Pewarisan method



Yang diwariskan adalah:
show(), minimize(), maximize()

Method hide() tidak diwariskan
Karena ditandai sebagai private

Pewarisan method



Method show() di-override di class Child

Super

- Keyword **super** dapat digunakan untuk memanggil **konstruktor** yang ada pada **superclass**.
- Penggunaan super dilakukan saat **inisialisasi** di dalam sebuah konstruktor kelas anak

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

Konstruktor Super Class

- Constructor dari **superclass** tidak dapat **dimatikan/dinon aktifkan**
- Kita dapat **memanfaatkan** konstruktor class induk untuk digunakan pada class anak
- Kita dapat mengakses method class induk yang bersifat public/protected dengan super
- Kita dapat mengakses atribut class induk yang bersifat public/protected dengan super

Subtype Substitution

- Jika *B* adalah turunan dari *A*, maka *B* dapat diinstansiasi dari class *B* atau bertipe class *A*, *tapi tidak sebaliknya!*
- Examples:

```
B obj = new B();      //ok

A obj = new B();      //ok

B obj = new A();      //tidak ok
```

Subtype Substitution

- Jika *B* adalah turunan dari *A,* maka *B* dapat diinstansiasi dari class *B* atau bertipe class *A*
- A a = new B()
- Obyek a akan memiliki konstruktor class B tapi dia akan memiliki method-method dari class A

Demo Inheritance

- Class Manusia
- Class Mahasiswa

Alur Pemanggilan Konstruktor

- **Dipanggil** dari kelas anak **terendah** (terkhusus)
- Dieksekusi mulai dari super class **tertinggi** (terumum) lalu menurun kepada class anak **terendah** (terkhusus)

Contoh Inheritance

```
class CivitasAkademika {
      public CivitasAkademika() {
        System.out.println("Semua warga universitas");
      }
}
 class Staff extends CivitasAkademika {
      public Staff() {
        System.out.println("yang mencari sesuap nasi");
      }
}
 public class StaffEdukatif extends Staff {
      public StaffEdukatif() {
        System.out.println("sebagai pengajar");
      }
      public static void m
        StaffEdukatif x = n
      }
}
```

```
Semua warga universitas
yang mencari sesuap nasi
sebagai pengajar
```

Final

- Class yang diberi modifier final **tidak dapat** diturunkan lagi
  - Error : “cannot inherit from final KelasFinal”
- Method yang diberi modifier final **tidak dapat dioverride** pada class anak
  - Contoh Error: “getX() in AnakFinal cannot override getX() in KelasFinal; *overridden method is final*”
- Variabel kelas yang diberi modifier final **dapat** diturunkan ke class anak
  - Diperlakukan sebagai k**onstanta**!

Review: Contoh class final

```
final class MyClass {

 public static void main (String[] args) {
    System.out.println ("hallo");
 }
}

class Coba extends MyClass{

}
```

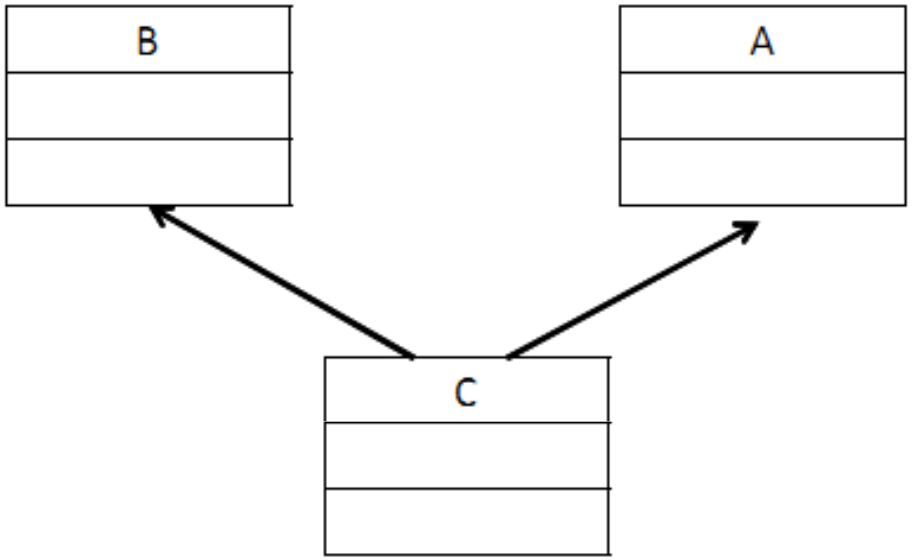
```
J:\arc\dosen\pbo\pertemuan3\MyClass.java:17: cannot inherit from final MyCla
class Coba extends MyClass{
                   ^
1 error
```

Review: Contoh method final

```
class MyClass {
 int x = 10;
 public final void setX(int x){
     this.x = x;
 }
 public int getX(){
     return x;
 }
}

class Coba extends MyClass{
 public void setX(int x){

 }
}
```

```
J:\arc\dosen\pbo\pertemuan3\MyClass.java:21: setX(int) in Coba cannot override setX(int) in MyClass; overridden method is final
 public void setX(int x){
             ^
1 error
```

Multiple Inheritance



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

Overriding

- Yang dioverride **bukan** method **static**
- *Memiliki jumlah, jenis, dan urutan parameter dan return yg sama (extactly)*
- Asesibilitas harus **minimal sama atau lebih besar**
- Method **final** tidak dapat dioverride

Demo Overriding

Conversion

- Mengubah suatu type data ke type data lain
- Terjadi **otomatis** jika terjadi perubahan jangkauan
- Ada 2 jenis:
  - **Narrowing** vs **Widening**

Wide vs Narrow

- Widening:
  - byte -> short -> int -> long -> float -> double
  - char -> int -> long -> float -> double
- Narrowing:

double -> float -> long -> int > char -> short > byte

Harus menggunakan operator (cast)

Misal :

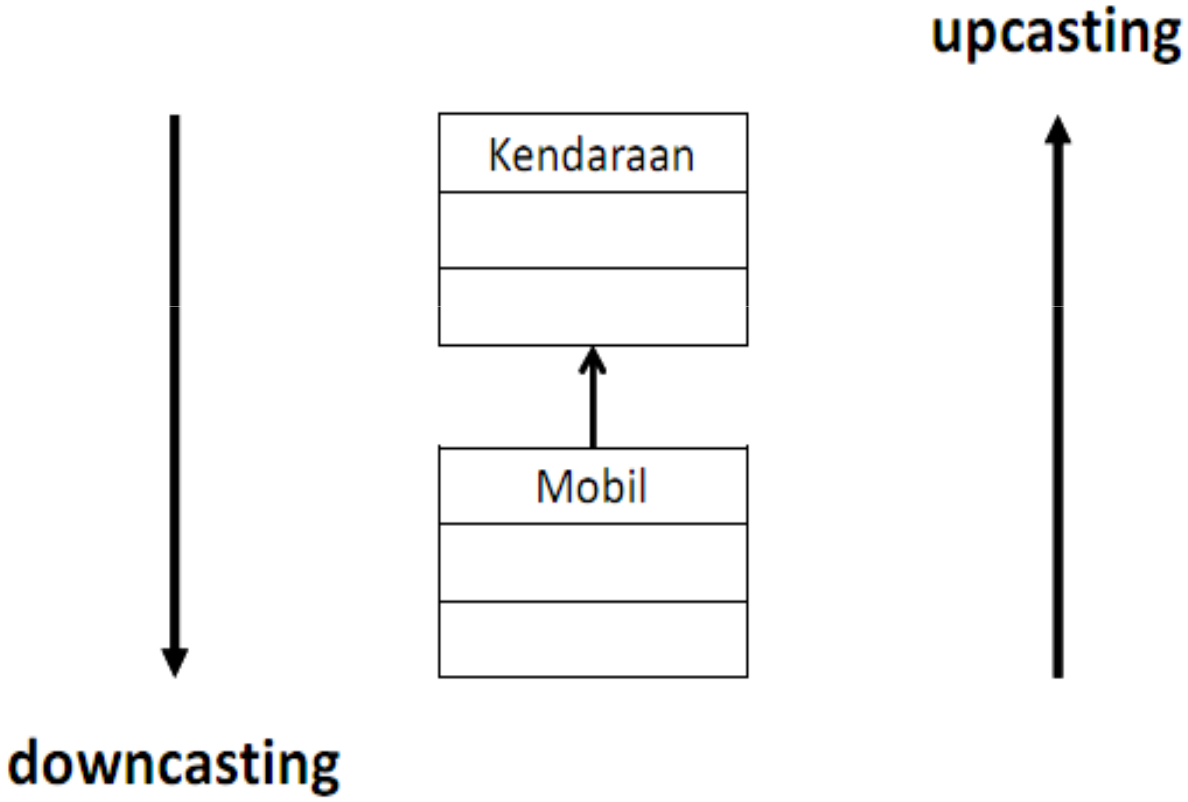
```
double data = 10.0;
float x = (float)(data);
```

Object casting

- Suatu object dengan type A bisa dikonversi menjadi object dengan type B dengan syarat :
  - Type A merupakan superclass dari type B
  - Type A merupakan subclass dari type B

- **Upcasting** dan **Downcasting**

Upcasting dan downcasting



Upcasting

- Terjadi secara **otomatis**
- Subclass memuat superclass, jadi bisa langsung di-’casting’ ke atas
- Upcasting dikatakan aman karena superclass lebih ‘umum’ ketimbang subclass (widening conversion)
- **Contoh**: jika Contoh class Cat adalah child dari class Animal
  - Jika a adalah obyek dari Animal dan c Cat, maka:
  - Animal a = (Animal) c; //contoh upcasting

Downcasting

- Harus menggunakan **operasi (cast)**
- Dalam banyak kasus akan menimbulkan **exception -> java.lang.ClassCastException**
- Untuk menghindari exception, bisa gunakan keyword **instanceof** untuk melakukan pengecekan terlebih dahulu
- **Contoh**: jika class Cat adalah child dari class Animal
  - Jika a adalah obyek dari Animal dan c Cat, maka:
  - Cat c = (Cat) a; //contoh downcasting

NEXT

- Abstract dan Interface