PEMROGRAMAN BERBASIS OBJECT

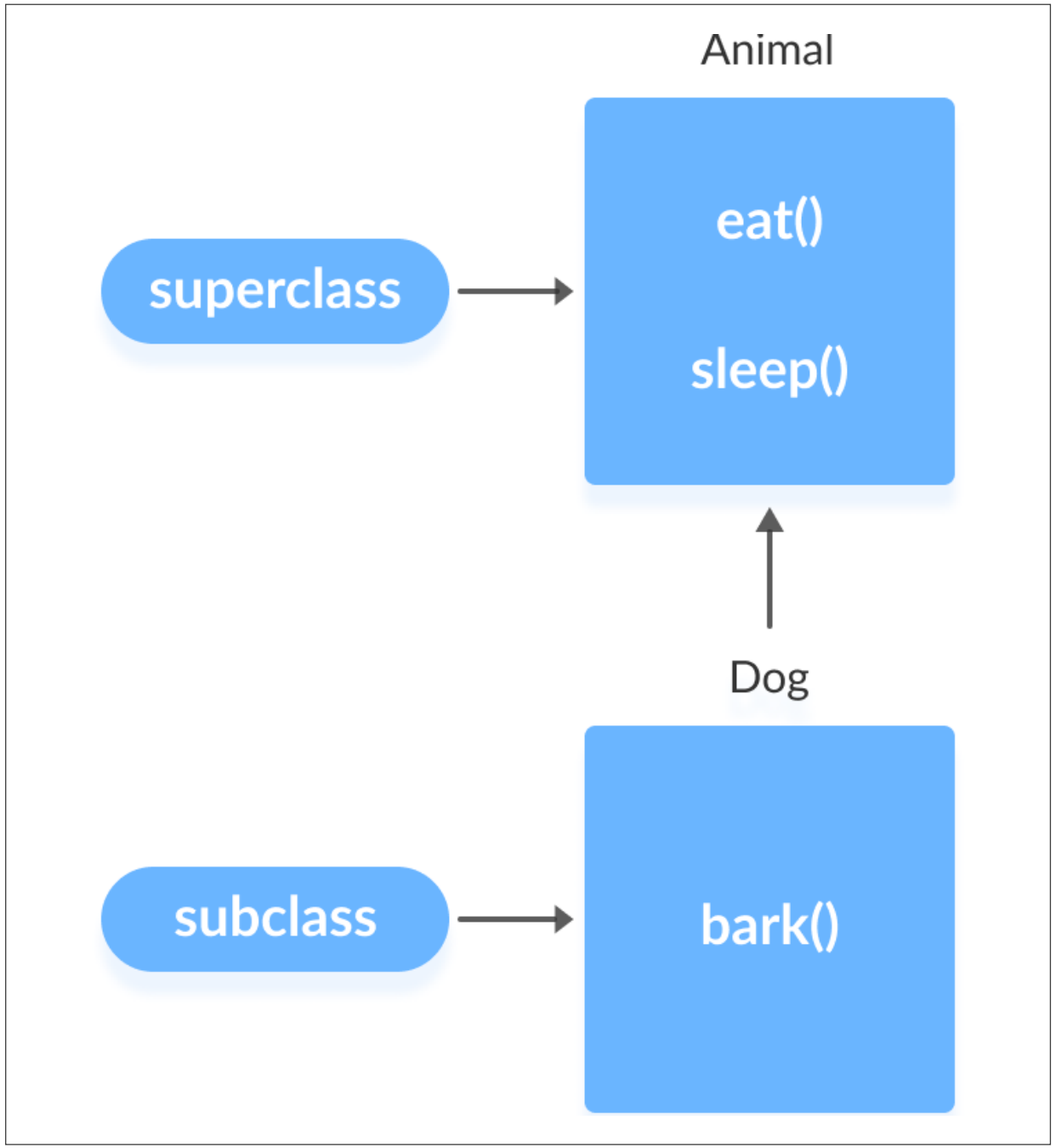
Minggu 6: **Inheritance**

Outline

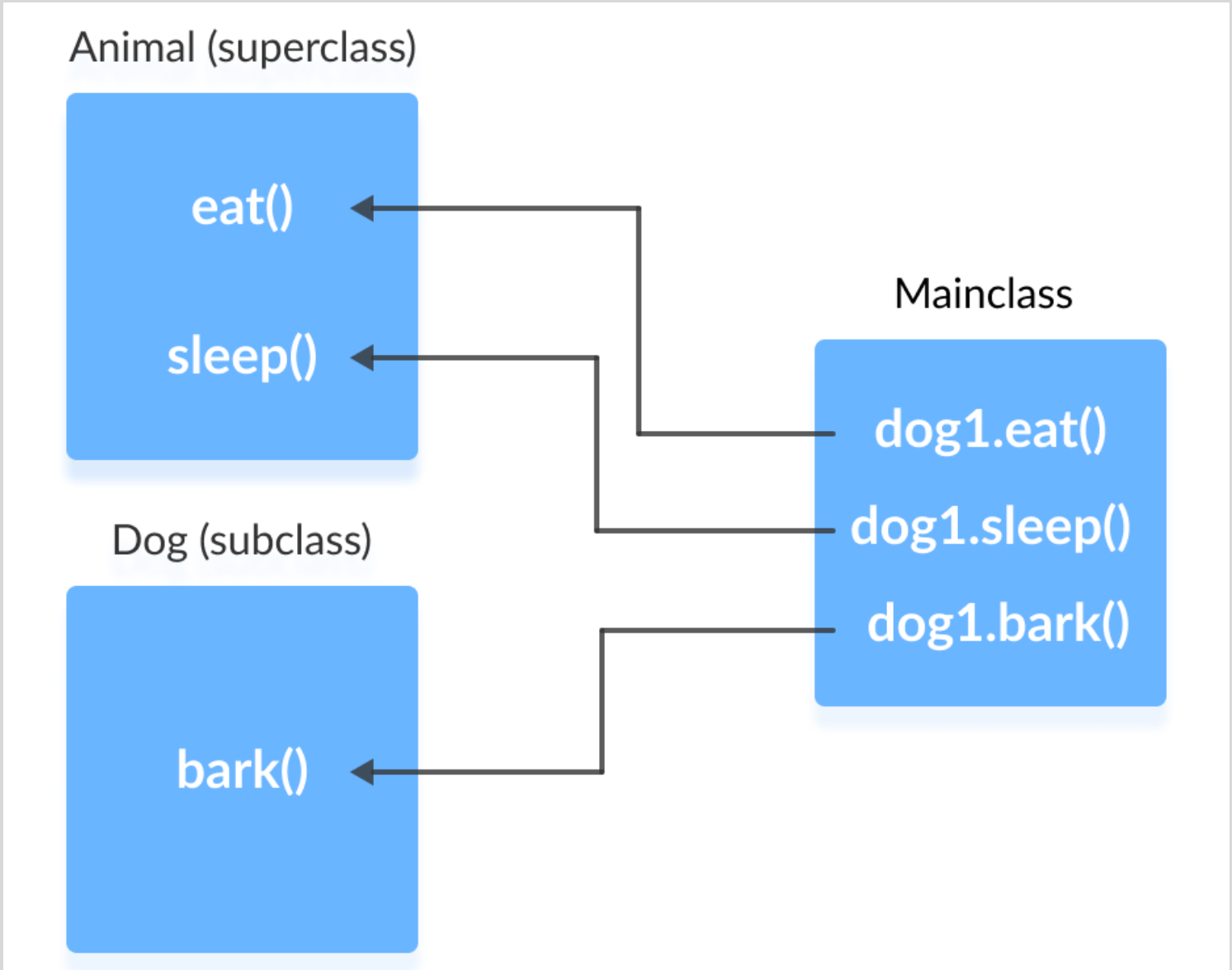
- Pengertian inheritance
- Deklarasi inheritance
- Access Control
- Super keyword
- Jenis inheritance

Inheritance

- Inheritance (Pewarisan) merupakan salah satu konsep dasar OOP.
- Konsep inheritance ini mengadopsi dunia nyata, dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan.
- Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.
- Konsep inheritance digunakan untuk memanfaatkan fitur 'code reuse' untuk menghindari duplikasi kode program



- Suatu class yang mempunyai class turunan dinamakan **superclass/parent class/base class**.
- Sedangkan class turunan itu sendiri seringkali disebut **subclass/child class/derived class**.
- Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class.


Animal (superclass)
eat()
sleep()
Dog (subclass)
bark()
Mainclass
dog1.eat()
dog1.sleep()
dog1.bark()

Deklarasi inheritance

- Dengan menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya.

```
public class <Subclass> extends <Superclass>
{
        …
}
```

Kapan kita menerapkan inheritance?

- Kita baru perlu menerapkan inheritance pada saat dijumpai ada suatu class yang dapat di-extend/diperluas/dikembangkan dari class lain

class Pegawai

```
public class Pegawai {
        public String nip;
        public String nama;
        public double gaji;
}
```

class Dosen

```
public class Dosen {
        public String nip;
        public String nama;
        public double gaji;
        public String NIDN;
}
```

- Dari 2 class di atas, kita lihat class **Dosen** memiliki atribut yang identik sama dengan class **Pegawai**, hanya saja ada tambahan atribut **NIDN (Nomor Induk Dosen Nasional)**.
- Sebenarnya yang terjadi disana adalah class Dosen merupakan **extension/perluasan** dari class Pegawai berupa tambahan atribut NIDN.
- Dengan konsep inheritance, deklarasi class Dosen dapat dituliskan sebagai berikut:

```
public class Dosen extends Pegawai {
        public String NIDN;
}
```

class Pegawai

```
public class Pegawai {
        public String nip;
        public String nama;
        public double gaji;
}
```

class Dosen

```
public class Dosen extends Pegawai {
        public String NIDN;
}
```

Kata Kunci “super”

- Kata kunci **super** dipakai untuk merujuk pada **parent class/object**
- Kata kunci **this** dipakai untuk merujuk pada **current class/object**
- Format penulisannya adalah sebagai berikut:
  - **super()**
    → merujuk pada constructor parent
  - **super.<namaAttribute>**
    → merujuk pada atribut dari parent
  - **super.<namaMethod>()**
    → merujuk pada method dari parent

Variable Scope

```java
public class Kendaraan {
    public int x = 5;
}
```

```java
public class Mobil extends Kendaraan{
    public int x = 10;

    public void displayInfo(int x){
        System.out.println("Nilai x: " + x);
        System.out.println("Nilai this.x: " + this.x);
        System.out.println("Nilai super.x: " + super.x);
    }
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        Mobil mobil1 = new Mobil();
        mobil1.displayInfo(12);
    }
}
```

```
Nilai x: 12
Nilai this.x: 10
Nilai super.x: 5
```

Variable Scope (2)

```java
public class Kendaraan {
    public int x = 5;
}
```

```java
public class Mobil extends Kendaraan{
    public int x = 10;

    public void displayInfo(){
        System.out.println("Nilai x: " + x);
        System.out.println("Nilai this.x: " + this.x);
        System.out.println("Nilai super.x: " + super.x);
    }
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        Mobil mobil1 = new Mobil();
        mobil1.displayInfo();
    }
}
```

```
Nilai x: 10
Nilai this.x: 10
Nilai super.x: 5
```

Variable Scope (3)

```java
public class Kendaraan {
    public int x = 5;
}
```

```java
public class Mobil extends Kendaraan {
    public String platNomor;

    public void displayInfo() {
        System.out.println("Nilai x: " + x);
        System.out.println("Nilai this.x: " + this.x);
        System.out.println("Nilai super.x: " + super.x);
    }
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        Mobil mobil1 = new Mobil();
        mobil1.displayInfo();
    }
}
```

```
Nilai x: 5
Nilai this.x: 5
Nilai super.x: 5
```

```java
public class Kendaraan {
    private int x = 5;

    public void displayInfo()
    {
        System.out.println("Nilai this.x: " + this.x);
    }
}
```

```java
public class Mobil extends Kendaraan{
    public int x = 10;

    public void displayInfo(int x){
        System.out.println("Nilai x: " + x);
        System.out.println("Nilai this.x: " + this.x);
        System.out.println("Nilai super.x: " + super.x);
    }
}
```

```java
public class Demo {
    Run | Debug
    public static void main(String[] args) {
        Kendaraan kendaraan1 = new Kendaraan();
        kendaraan1.displayInfo();
    }
}
```

```
Nilai this.x: 5
```

Kesimpulan

- **x**

  → merujuk pada x terdekat, cari dulu variable/parameter. Jika tidak ada berarti merujuk pada atribut class tersebut. Jika tidak ada berarti merujuk pada atribut parent class
- **this.x**

  → merujuk pada atribut x pada current class/object. Jika tidak ada berarti merujuk pada atribut parent class
- **super.x**

  → merujuk pada atribut x pada parent class/object

Access Control

- Dalam dunia nyata, suatu entitas induk bisa saja tidak mewariskan sebagian dari apa-apa yang ia miliki kepada entitas turunan karena sesuatu hal.
- Demikian juga dengan konsep inheritance dalam OOP.
- Suatu parent class bisa saja **tidak mewariskan** sebagian atribut atau method kepada subclass-nya. Hal ini dilakukan dengan memberikan access level modifier **private**

Access Control

```java
public class Kendaraan {
    private int x = 5;

    public void displayInfo()
    {
        System.out.println("Nilai this.x: " + this.x);
    }
}
```

```java
public class Mobil extends Kendaraan{
    public int x = 10;

    public void displayInfo(int x){
        System.out.println("Nilai x: " + x);
        System.out.println("Nilai this.x: " + this.x);
        System.out.println("Nilai super.x: " + super.x);
    }
}
```

```
The field Kendaraan.x is not visible Java(33554503)
int x
View Problem (Alt+F8)    Quick Fix... (Ctrl+.)
```

Single Inheritance

- Konsep inheritance dengan sublass yang mempunyai satu parent class.

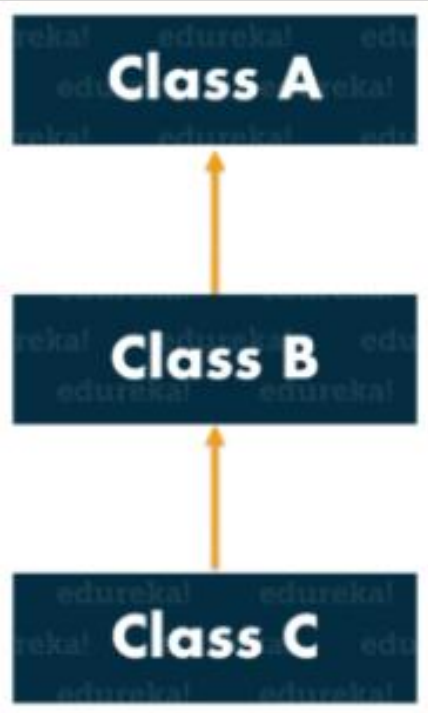

```
class Animal {
      void eat() {
            System.out.println("eating...");
      }
}

class Dog extends Animal {
      void bark() {
           System.out.println("barking...");
      }
}

class TestInheritance{
    public static void main(String args[]) {
            Dog d=new Dog();
            d.bark();
            d.eat();
      }
}
```

Multilevel Inheritance

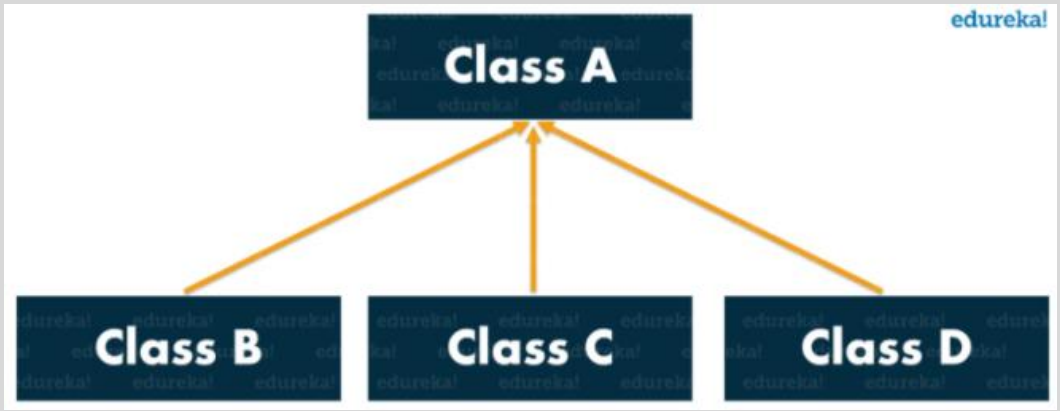
- Ketika kelas diturunkan dari kelas yang juga diturunkan dari kelas lain, yaitu kelas yang memiliki lebih dari satu kelas induk tetapi pada tingkat yang berbeda




```
class Animal{
    void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
    void bark(){System.out.println("barking...");}
}
class Puppy extends Dog{
    void weep(){System.out.println("weeping...");}
}
class TestInheritance2{
    public static void main(String args[]){
        Puppy puppy1=new BabyDog();
        puppy1.weep();
        puppy1.bark();
        puppy1.eat();
    }
}
```

Hierarchical Inheritance

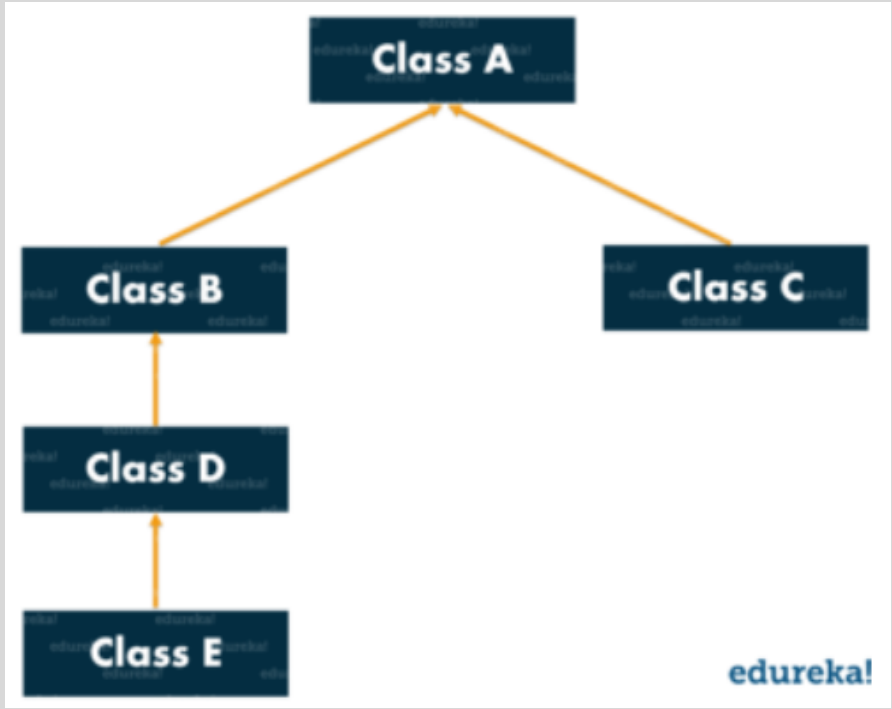
- Ketika sebuah kelas memiliki lebih dari satu kelas turunan (*subclass*) atau dengan kata lain, lebih dari satu kelas turunan memiliki kelas induk yang sama.



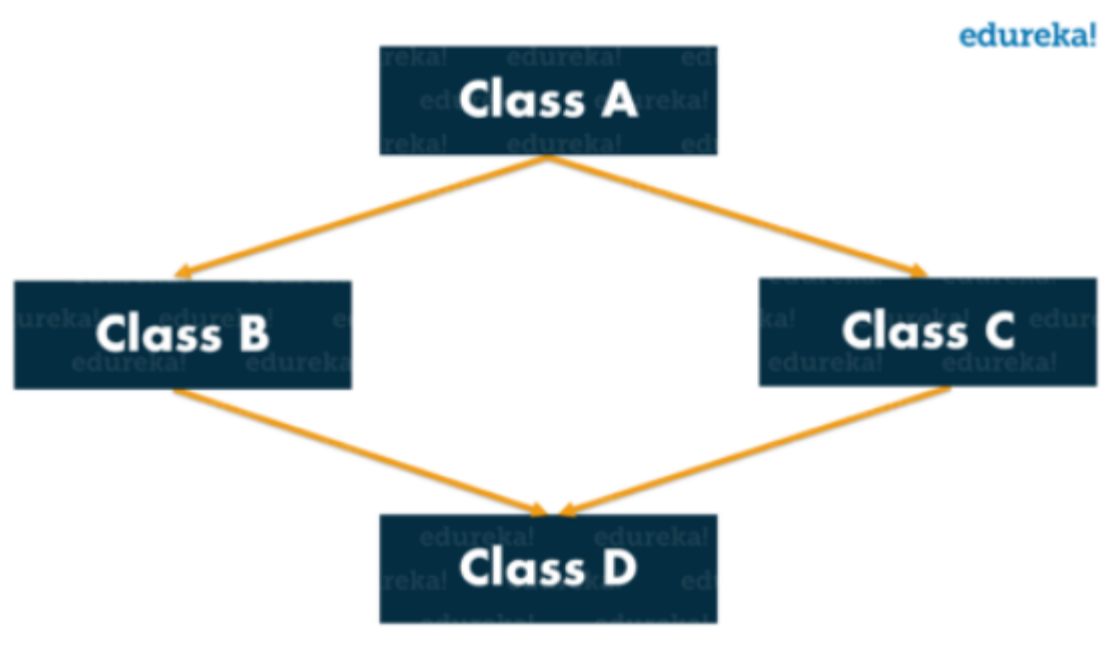

```
class Animal{
   void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
   void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
   void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
    public static void main(String args[]){
            Cat cat1=new Cat();
            cat1.meow();
            cat1.eat();
            //cat1.bark(); //C.T.Error
    }
}
```

Hybrid Inheritance

- Kombinasi dua atau lebih jenis inheritance.



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.

Mengapa Multiple Inheritance tidak bisa diimplementasikan di Java?

```
class B extends D{
   void show(){
       System.out.println("Welcome");
   }
}

class C extends D{
   void show(){
        System.out.println("Good Morning");
   }
}
```

```
class A extends B,C{
    void print(){System.out.println("This is multiple inheritance");}
}

class TestInheritance4{
     public static void main(String args[]){
            A obj=new A();
            obj.print();
            obj.show(); //method show() mana yang akan dipanggil
      }
}
```

Tugas

- Buatlah sebuah contoh hierarchical inheritance dengan 1 parent class dan 2 child class. Pada parent class terdapat minimal 3 atribut dan pada setiap child class terdapat minimal 1 atribut tambahan. Buatlah class diagramnya

Best Practice

Kendaraan
kecepatan: int

Mobil
warna: String

Becak
warna: String

Motor
warna: String

TERIMA KASIH