Pewarisan Jamak

Brigida Arie Minartiningtyas, M.Kom.

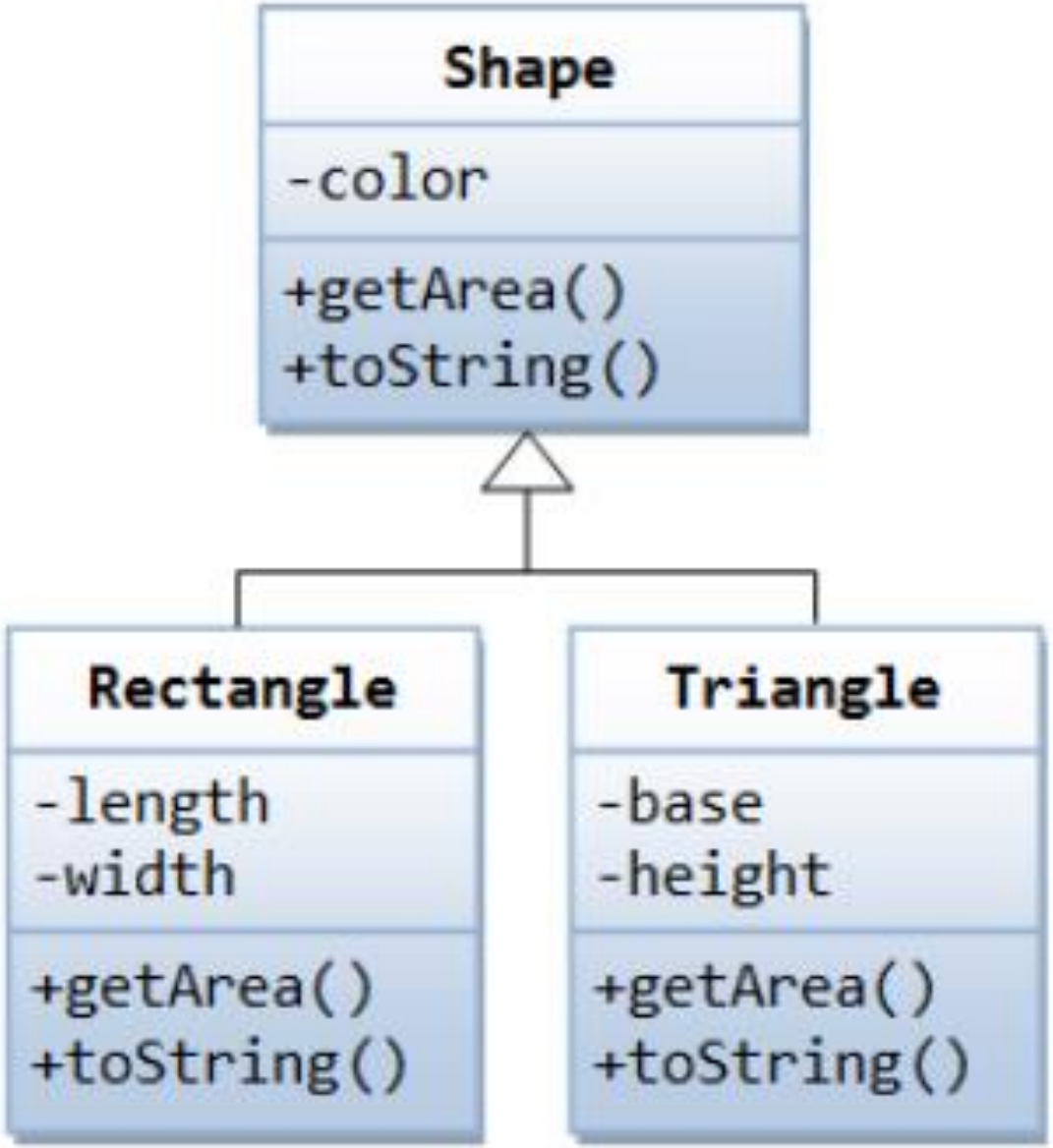
Polimorfisme Statik

- Diwujudkan dengan menggunakan overloading function, operator atau constructor
- Binding Method dilakukan secara statik
- Ciri :
  - Nama method sama
  - Parameter berbeda (tipe parameter dan/atau jumlah paramater dan/atau urutan parameter)
  - Tipe balikan tidak merupakan kriteria

Polimorfisme Dinamik

- Overridding Method
- Interface
- **Abstract Class di Java**

Abstract Class



Pendefinisian kelas induk yang di dalamnya terdapat deklarasi method yang tidak memerlukan implementasi sama sekali

Kelas Abstrak

```
abstract class nama-kelas{
      badan kelas
}
```

Method Abstract

```
abstract tipe namaMethod (daftar-parameter)
```

```
abstract public class Shape {
    ......
    public abstract double getArea();
    public abstract void draw();
}
```


"Now Speak!"
"Woof"
"Quack"
"Meow"

```
abstract public class Animal {
    abstract public void greeting();
}
```

```java
public class Cat extends Animal {
    @Override
    public void greeting() {
        System.out.println("Meow!");
    }
}
```

```java
public class Dog extends Animal {
    @Override
    public void greeting() {
        System.out.println("Woof!");
    }

    public void greeting(Dog another) {
        System.out.println("Wooooooooooof!");
    }
}
```

```java
public class BigDog extends Dog {
    @Override
    public void greeting() {
        System.out.println("Woow!");
    }

    @Override
    public void greeting(Dog another) {
        System.out.println("Wooooooowwwwww!");
    }
}
```

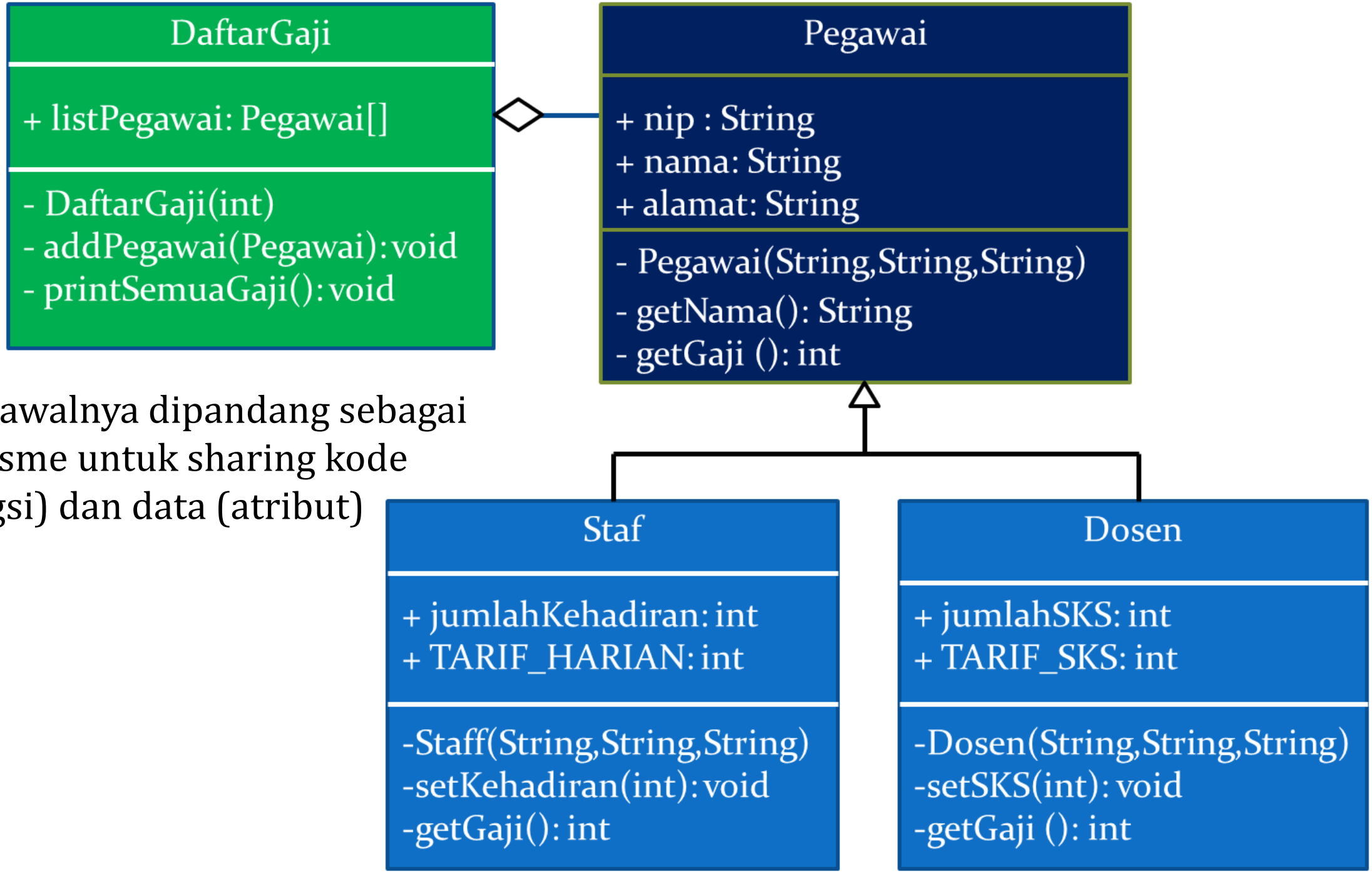
```java
public class TestAnimal {
    public static void main(String[] args) {
        // Using the subclasses
        Cat cat1 = new Cat();
        cat1.greeting();
        Dog dog1 = new Dog();
        dog1.greeting();
        BigDog bigDog1 = new BigDog();
        bigDog1.greeting();
```

Polimorfisme Statik

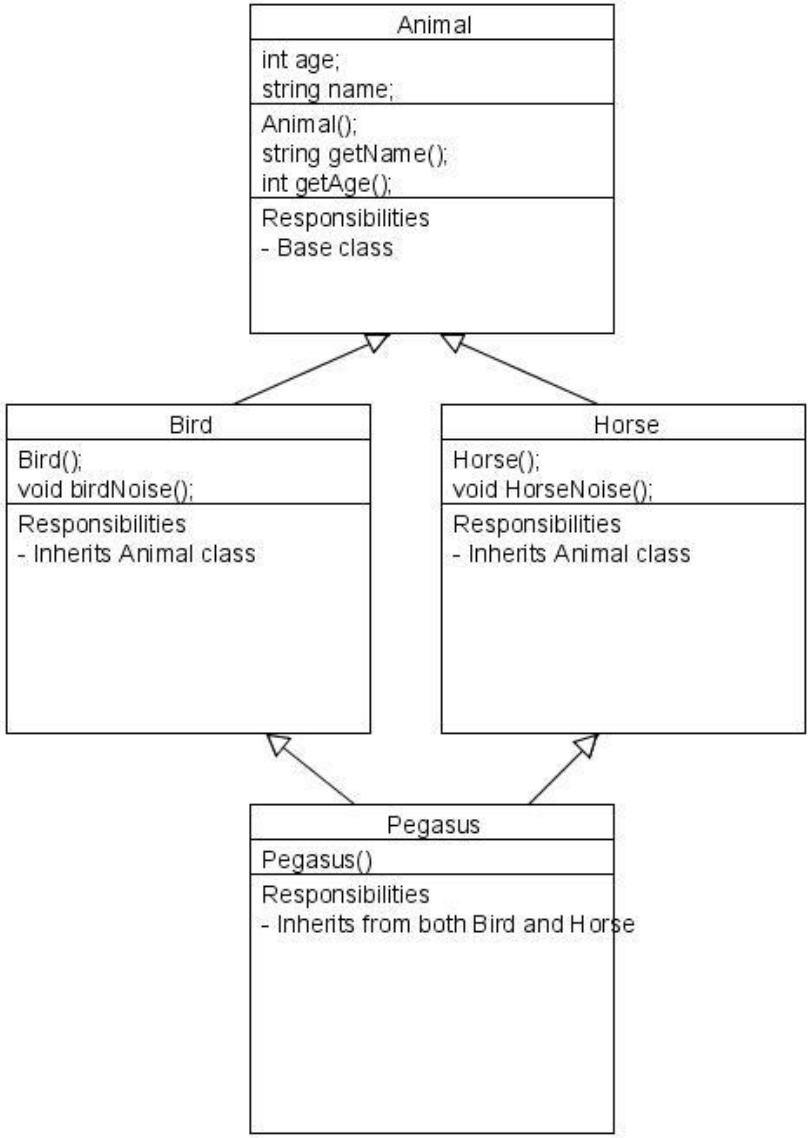
- Diwujudkan dengan menggunakan overloading function, operator atau constructor
- Binding Method dilakukan secara statik
- Ciri :
  - Nama method sama
  - Parameter berbeda (tipe parameter dan/atau jumlah paramater dan/atau urutan parameter)
  - Tipe balikan tidak merupakan kriteria

Polimorfisme Dinamik

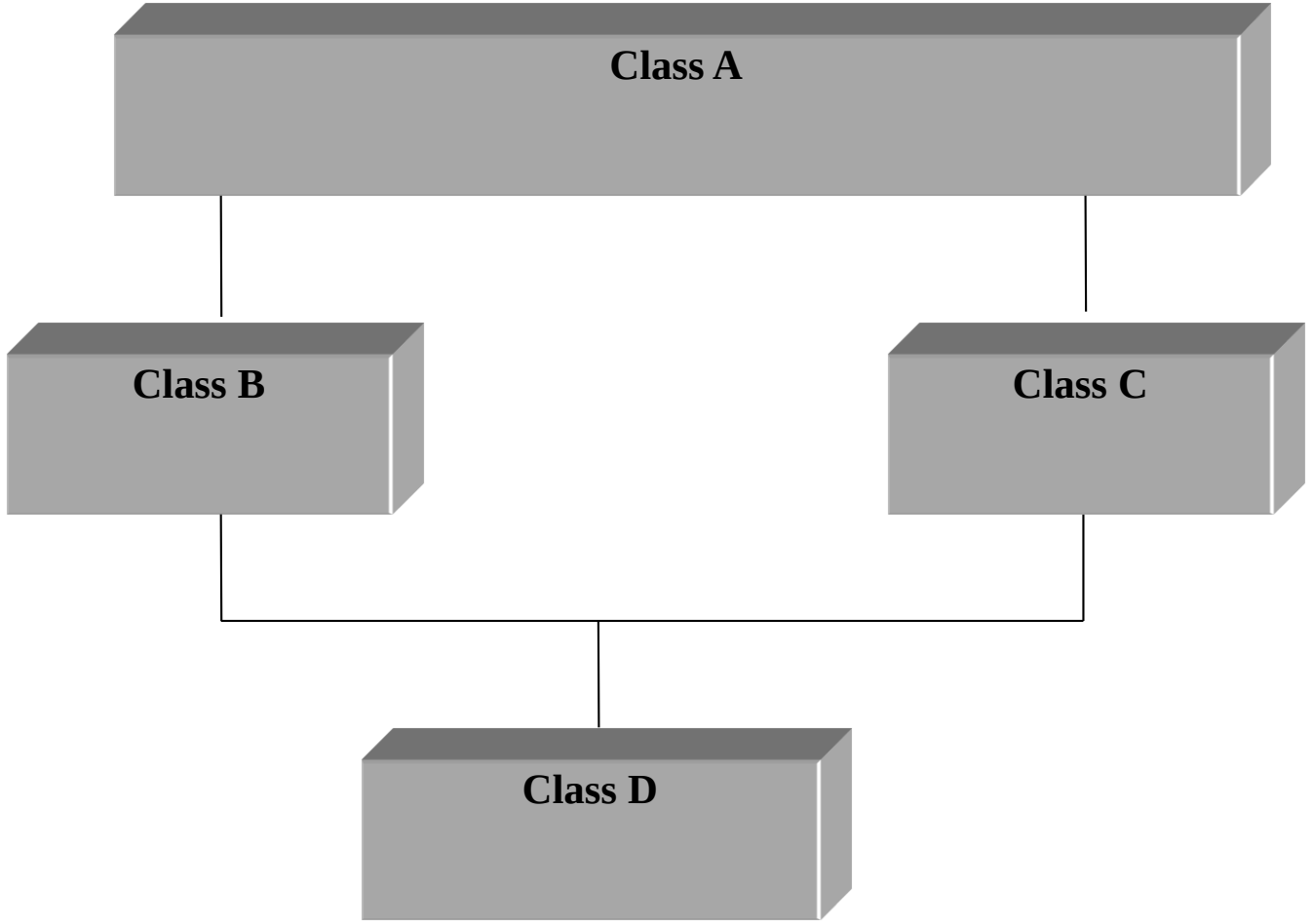
- Overridding Method
- **Interface**
- Abstract Class di Java



Pewarisan awalnya dipandang sebagai mekanisme untuk sharing kode (fungsi) dan data (atribut)



Pewarisan Jamak (multiple inheritance) dipandang sebagai mekanisme untuk membentuk suatu sub kelas dari beberapa implementasi super kelas



Problem – Ambiguitas

1. Timbul ketika 2 kelas induk memiliki suatu fungsi dengan nama yang sama.
2. Kelas turunan memiliki beberapa salinan kelas dasar yang sama.

Java **TIDAK** mendukung pewarisan Jamak

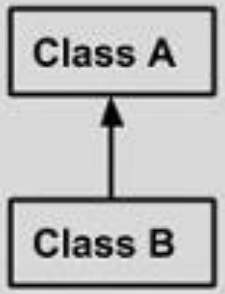
Java **HANYA** memiliki *pewarisan tunggal*

- Suatu kelas turunan hanya memiliki ***satu kelas induk***

Lalu bagaimana perwujudannya dalam Java ?

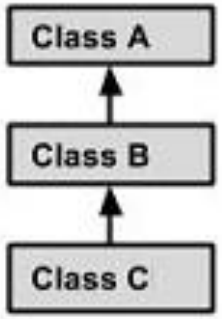
- Java mewujudkannya dengan menggunakan *Interface*
- **Multiple Interface Inheritance** **BUKAN** Multiple Implementation Inheritance

Single Inheritance



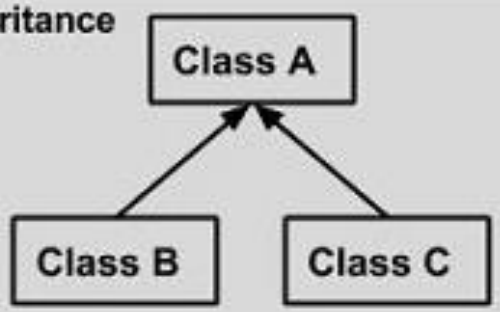

```
public class A {
   .......
}
public class B extends A {
   .........
}
```

Multi Level Inheritance



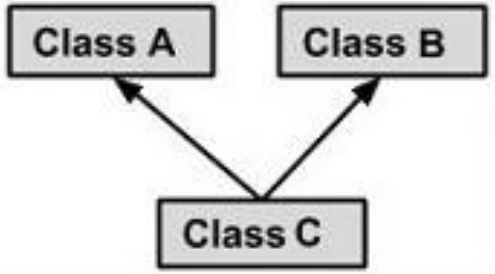

```
public class A { ...................}

public class B extends A {...................}

public class C extends  B {..................... }
```

Hierarchical Inheritance



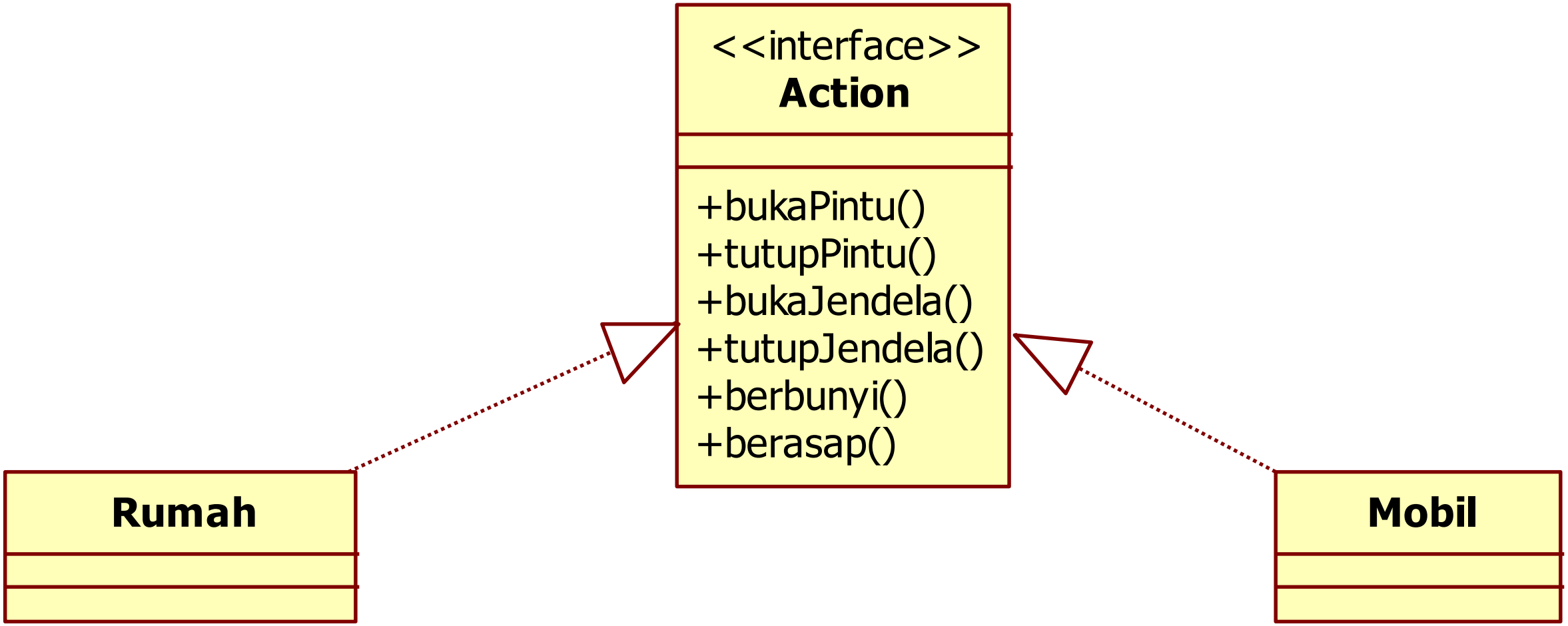

```
public class A { ...................}

public class B extends A {...................}

public class C extends A {..................... }
```

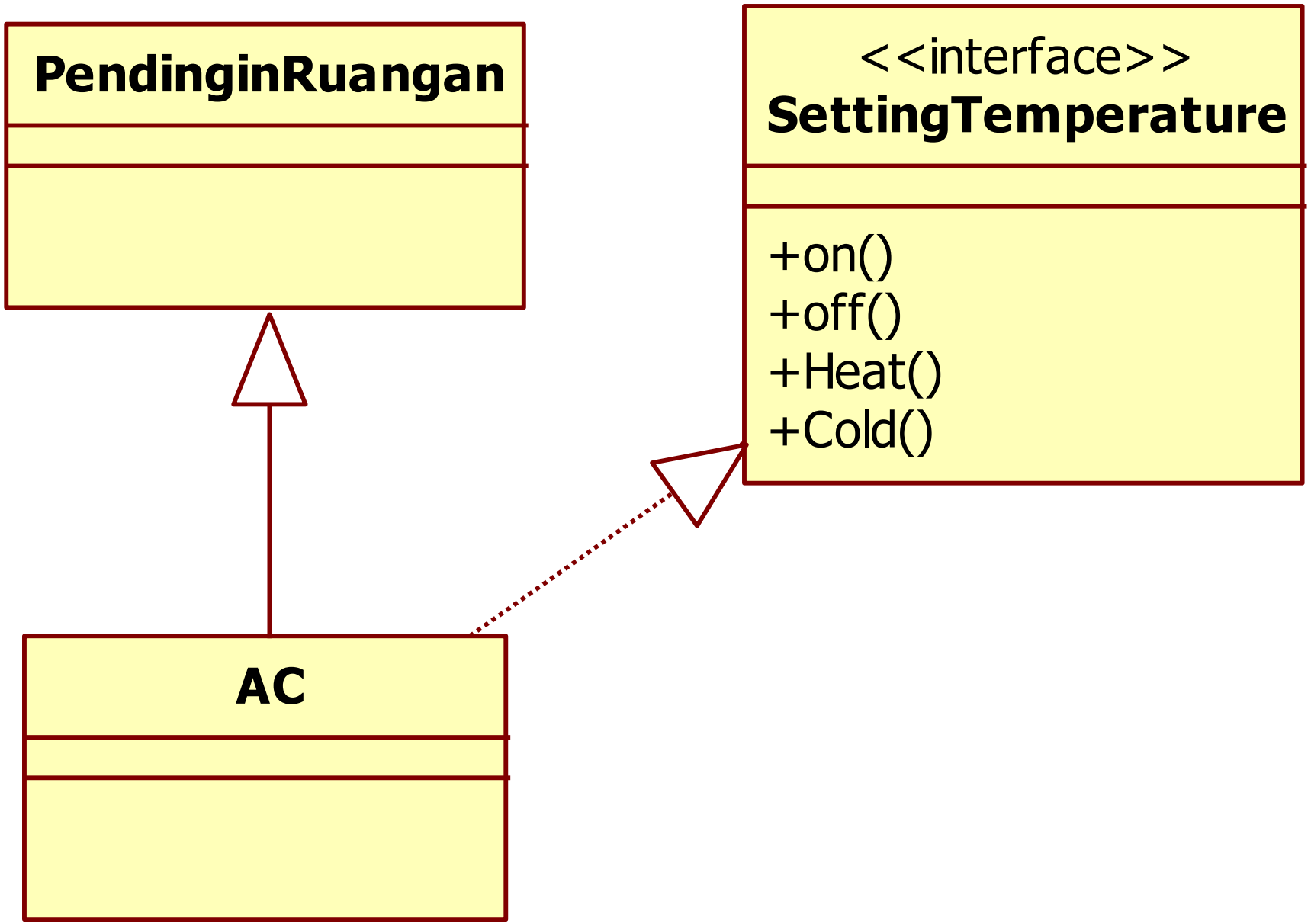
Multiple Inheritance

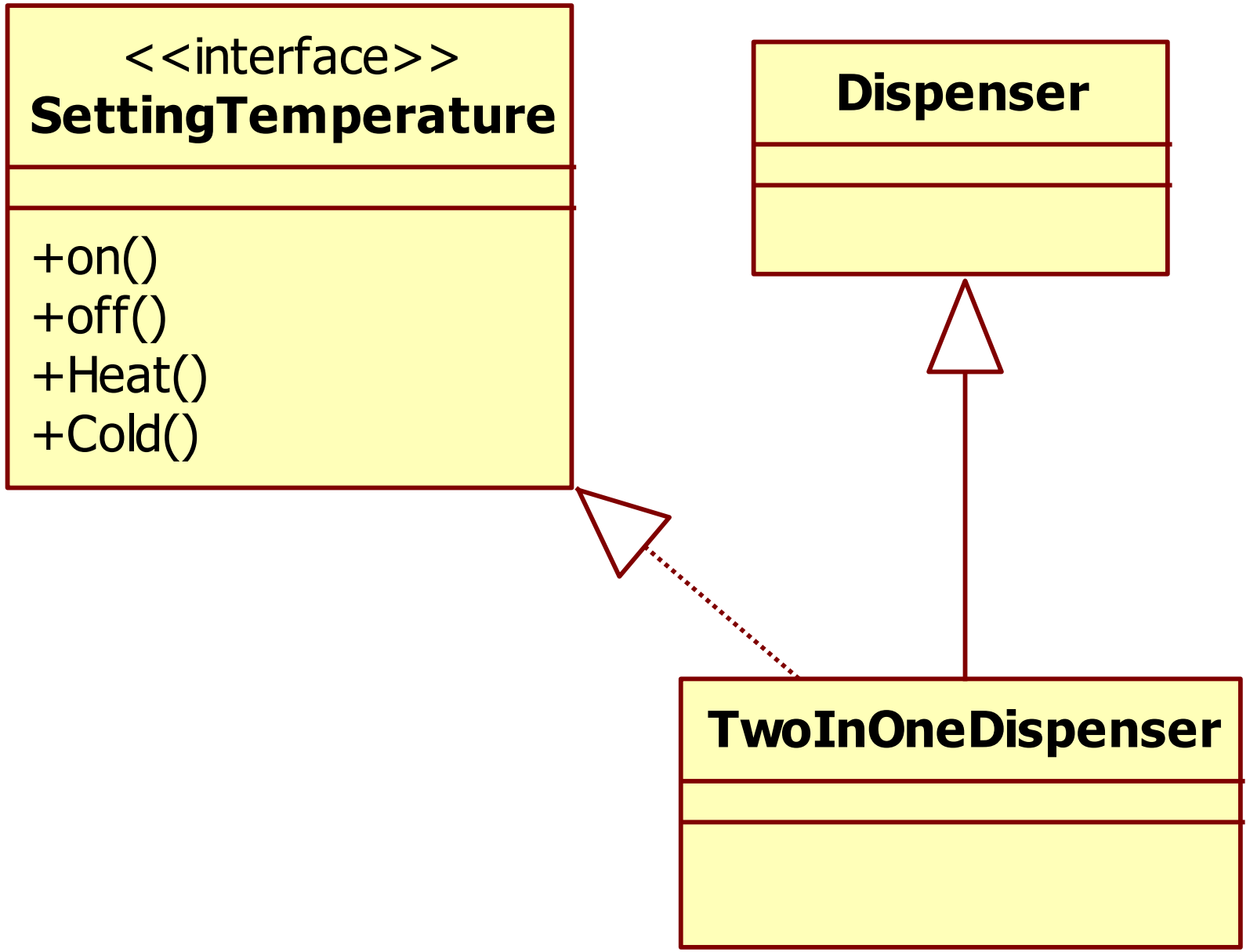


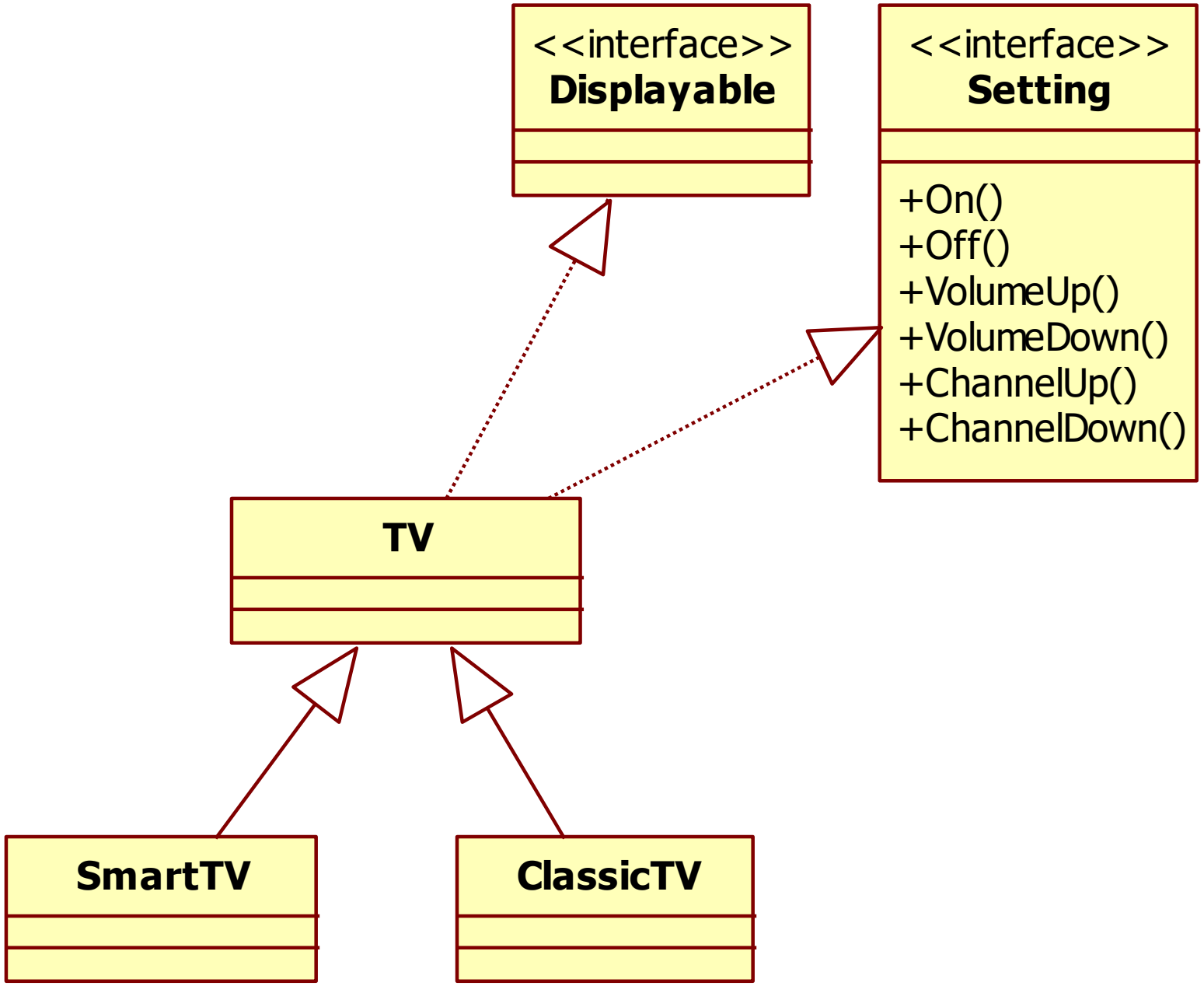

```
public class A { ...................}

public class B {...................}

public class C extends  A,B {
   ....................
} // Java does not support mutiple Inheritance
```

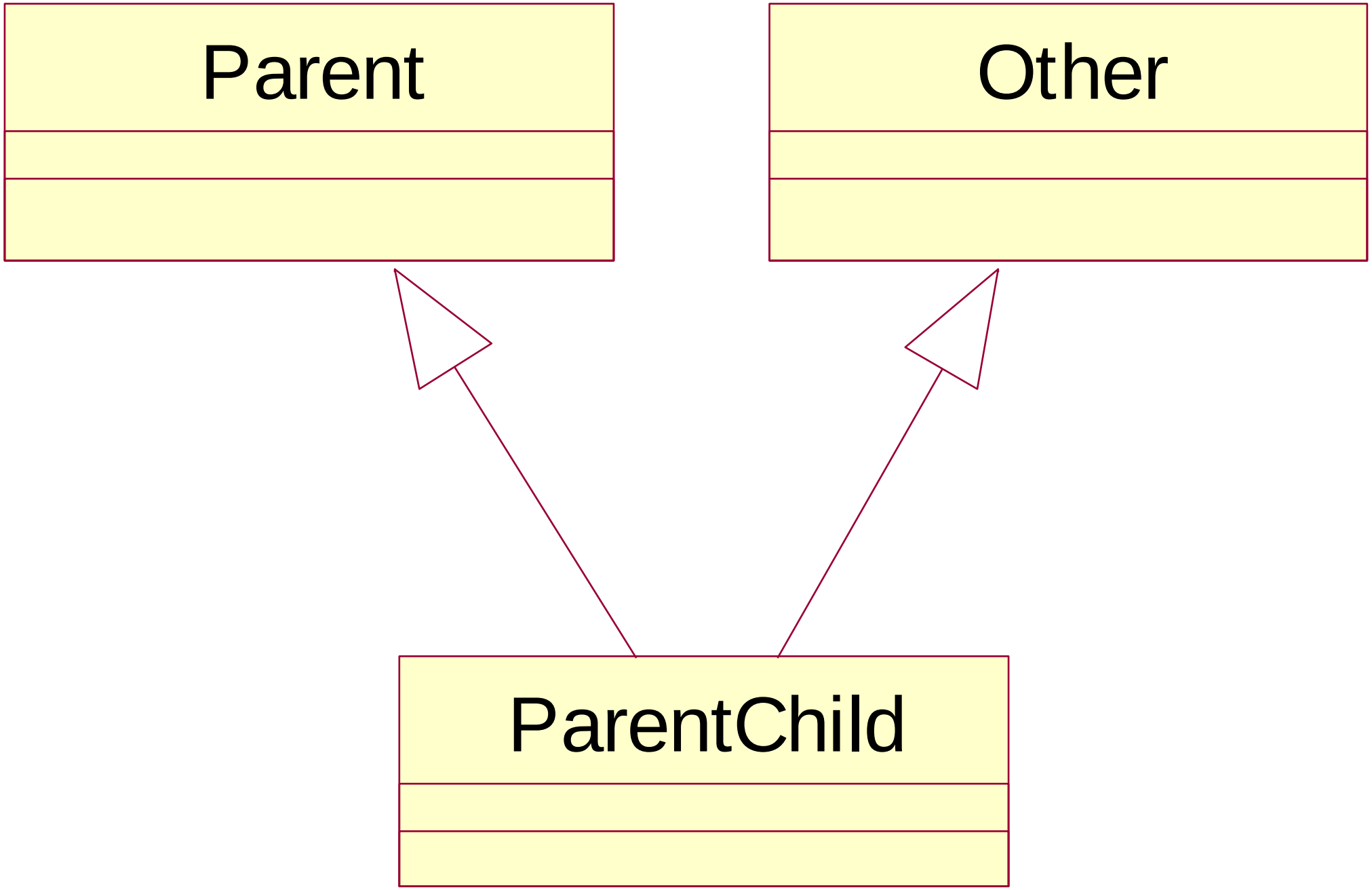
INTERFACE

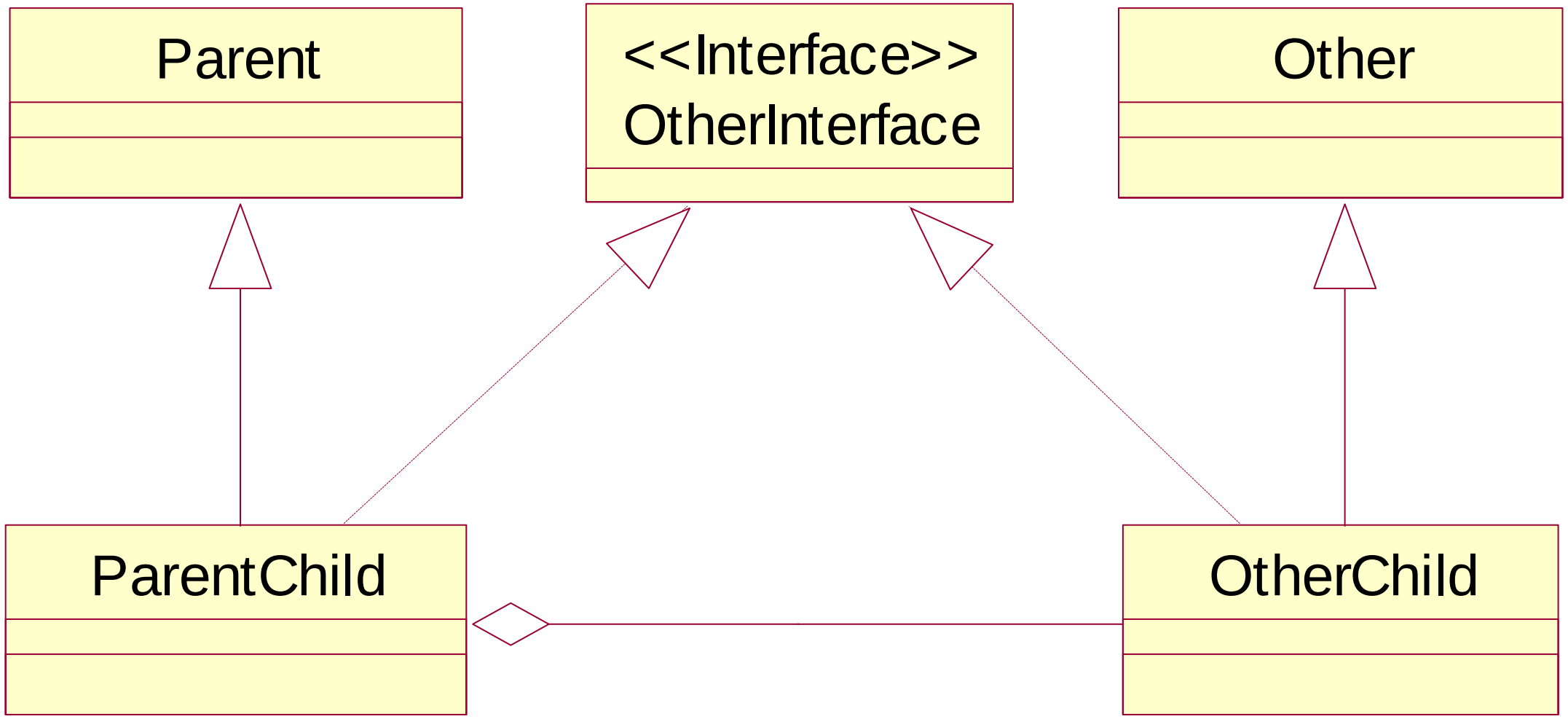

<<interface>>
Action
+bukaPintu()
+tutupPintu()
+bukaJendela()
+tutupJendela()
+berbunyi()
+berasap()
Rumah
Mobil


PendinginRuangan
<<interface>>
SettingTemperature
+on()
+off()
+Heat()
+Cold()
AC


<<interface>>
SettingTemperature
+on()
+off()
+Heat()
+Cold()
Dispenser
TwoInOneDispenser


<<interface>>
Displayable
<<interface>>
Setting
+On()
+Off()
+VolumeUp()
+VolumeDown()
+ChannelUp()
+ChannelDown()
TV
SmartTV
ClassicTV


Parent
Other
ParentChild


Parent
<<Interface>>
OtherInterface
Other
ParentChild
OtherChild

Kumpulan konstanta dan method yang tidak memiliki implementasi

- Interface seperti kontrak dengan ikatan implements

Semua method dalam interface adalah abstract.

- Tapi pada pendeklarasiannya tidak perlu menggunakan keyword abstract.

Interface digunakan untuk mendukung multiple inheritance (satu class memiliki superclass lebih dari satu)

```
Interface namaInterface {
    deklarasiInterface
  }
```

```
public interface Shape {

    double volume();

    double area();

}
```

Interface bernama Shape

Method yang tidak berisi Implementasi

Seluruhnya bersifat public abstract
(tidak dapat diset private, protected atau *default*)

Selain bisa mendeklarasikan method abstract, di dalam interface juga dapat diberikan attribute final (konstanta)

Konstanta ini juga diwariskan kepada class yang mengimplementasikan interface tersebut

```
public interface mp3Player {
    public static final int STATUS; //final dan static
    List TRACKLIST;
    void playTrack();
    void stopTrack();
    void volumeUp();
    void volumeDown();
}
```

Suatu class dapat meng-implementasikan lebih dari satu interface

Class yang mengimplementasikan suatu interface harus membuat implementasi dari method-method yang ada pada interface

Seluruh method pada interface secara otomatis bersifat public, method pada class yang mengimplementasikan interface harus diset ke public

```
public class Persegi implements Shape {
        private int sisi;

        public double volume() {
                return 0;
        }

        public double area() {
                return sisi * sisi;
        }
}
```

```
public interface kamera
{
        public void setPixel(float pixel);
        public void ambilGambar();
}
```

Fungsinya adalah membuat suatu class yang bisa diimplementasikan oleh berbagai class lain bahkan yang tidak berelasi sama sekali

Interface dapat menerima warisan dari interface lain (bisa lebih dari satu)

Class bisa mengimplements lebih dari satu interface

Class yang mengimplementasikan interface harus mengimplementasikan semua method interface

- Jika tidak, maka class tersebut harus dideklarasikan sebagai sebuah abstract class

Interface

```
public interface NamaInterface
extends InterfaceA, InterfaceB{

...

}
```

Implements

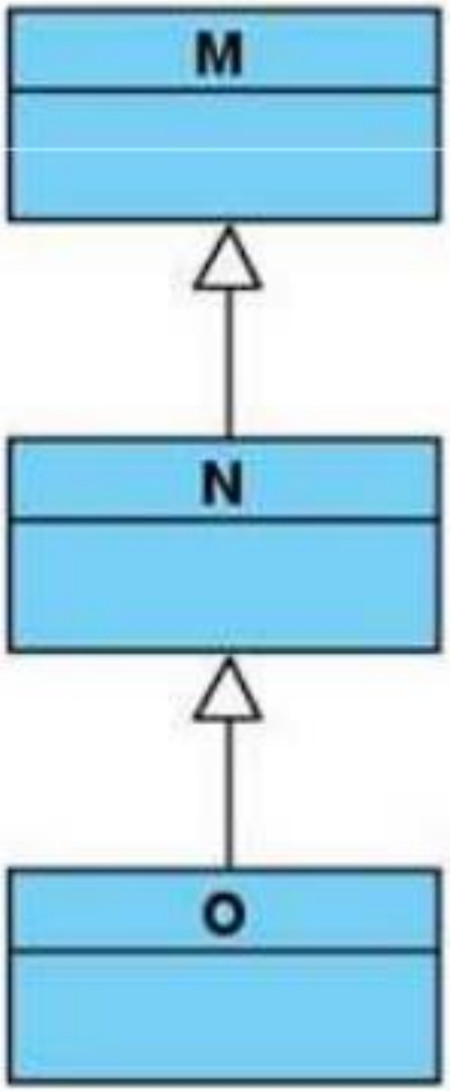
```
class namaKelas [extends namaKelasSuper]
implements NamaInterface1, NamaInterface2{

    …

    …

    …

}
```

Interface

- Keyword : Implements
- Semua method abstract
- Tidak dapat mewarisi abstract class
- Hanya dapat mendeklarasikan konstanta

Abstract

- Keyword : Extends
- Hanya beberapa atau mungkin hanya satu method yang abstract
- Bisa mengimplementasikan Interface
- Dapat mendeklarasikan variable instance

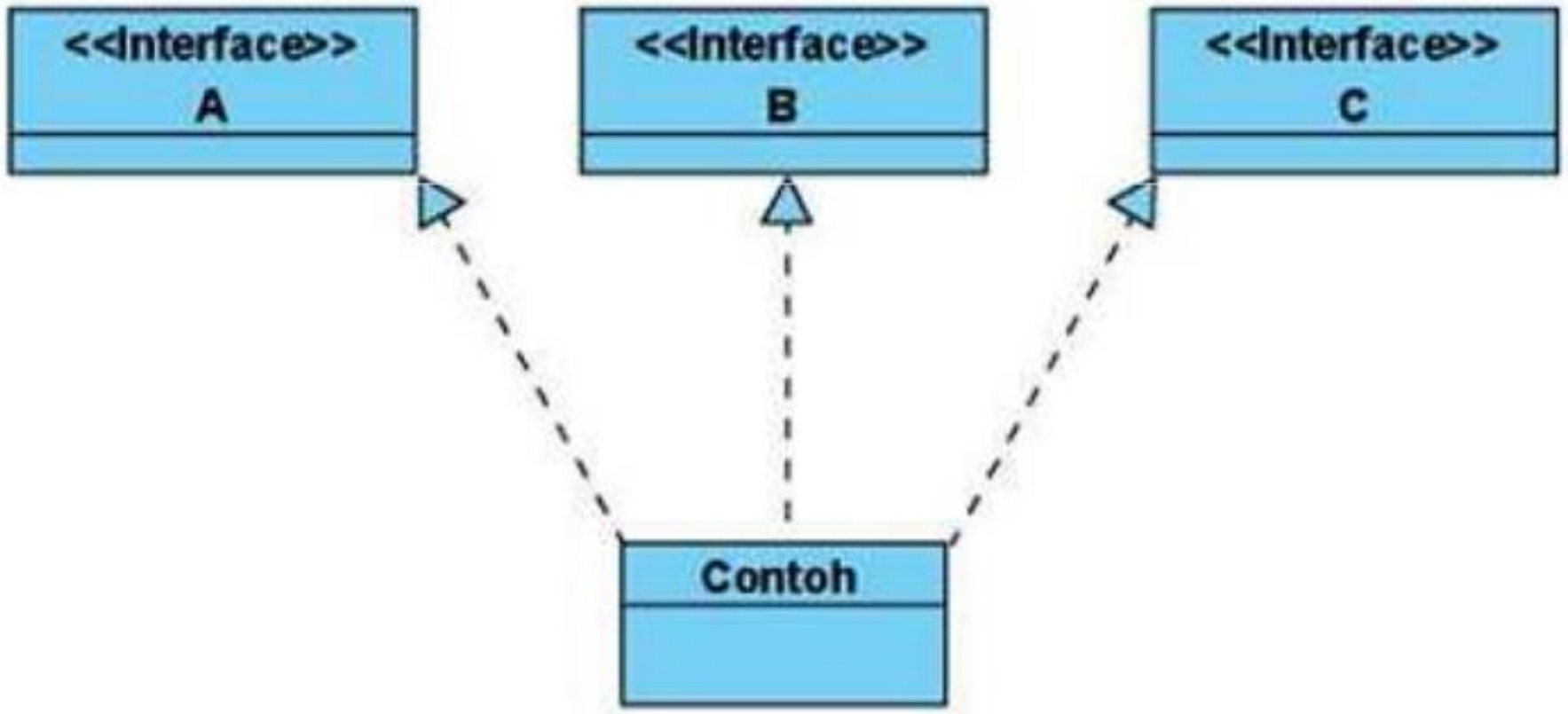


class M { ... }

class N extends M { ... }

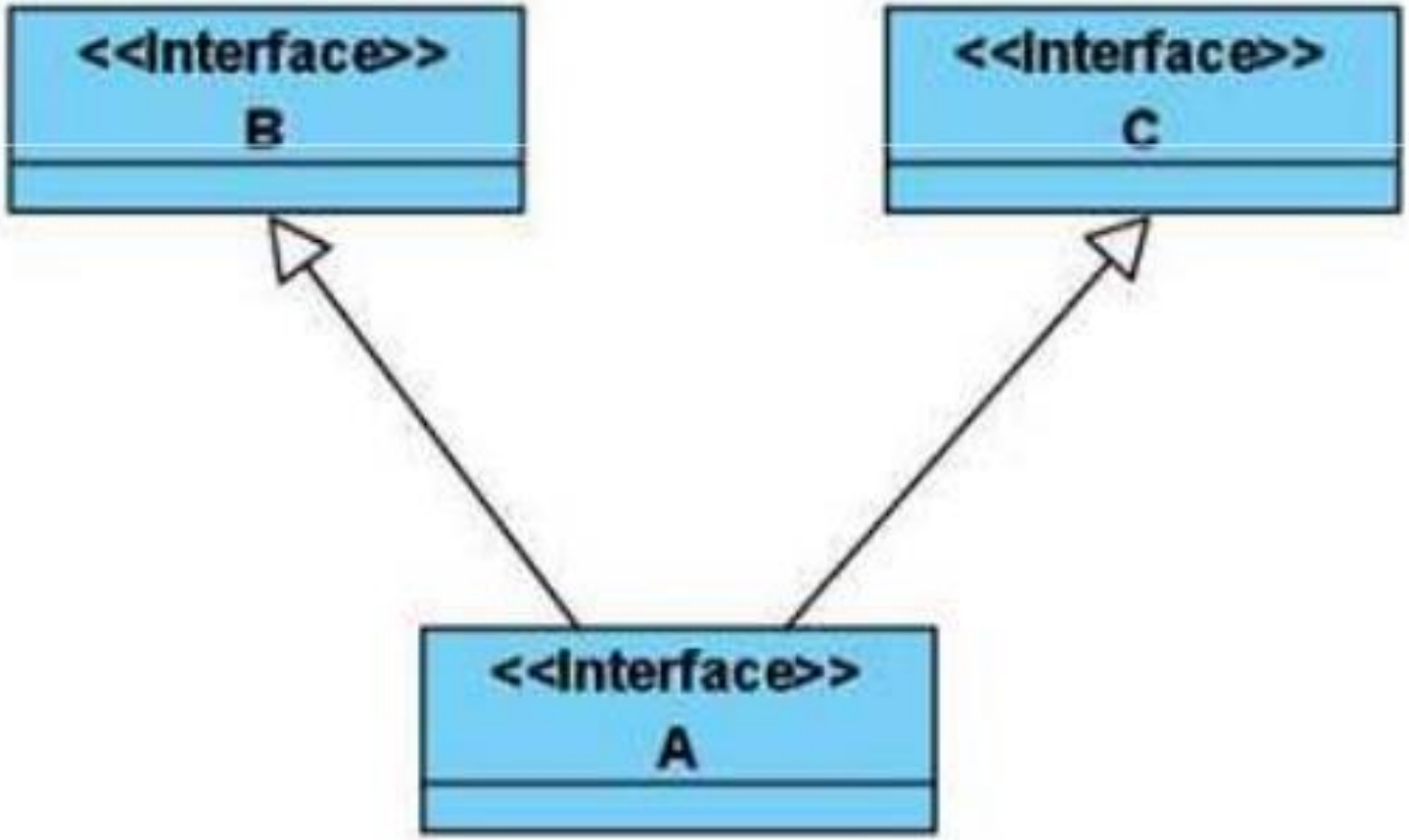
class O extends N { ... }

Suatu class hanya bisa meng-extends satu class lainnya



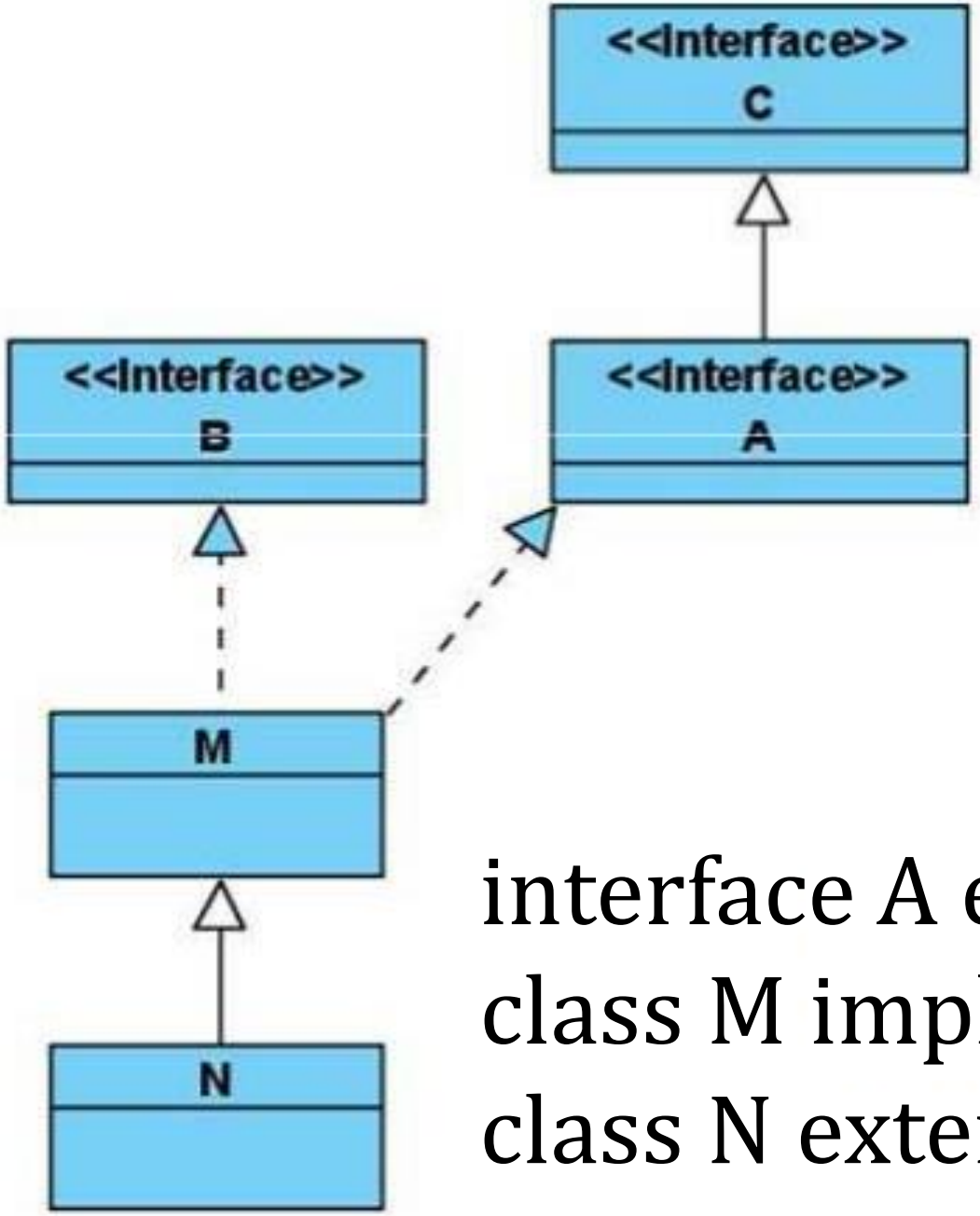
class Contoh implements A, B, C { ... }

Satu class bisa meng-implements lebih dari satu interface



interface A extends B, C { ... }

Suatu interface dapat meng-extends lebih dari satu interface lainnya



interface A extends C
class M implements B, A
class N extends M

```
class Parent
{
        private int val;

        public Parent(int value) {
          this.val = value;}
        public int getValue() {
          return this.val;}
        // some code omitted
}
```

```
class Other
{
        private int val;
        public Other(int value) {this.val = val;}

        public void whatever()
        {System.out.println("whatever methods \n");}
}
```

```
interface OtherInterface
{void whatever();}

class OtherChild extends Other implements OtherInterface
{public OtherChild (int value){
     super(value);
     }
}

class ParentChild extends Parent implements OtherInterface
{
     private OtherInterface child;
     public ParentChild(int val) {
          super(val);
          child = new OtherChild(val);
        }
     public void whatever() {child.whatever();}
}
```