UNIVERSITAS TEKNOLOGI SUMBAWA

Pemrograman Berorientasi Objek

Prodi Informatika – Fakultas Rekayasa Sistem

Semester Ganjil, 2024/2025

Oleh : I Made Widiarta

Agenda

- Inheritance

UNIVERSITAS TEKNOLOGI SUMBAWA

Inheritance

Inheritance

UNIVERSITAS TEKNOLOGI SUMBAWA

- Inheritance atau pewarisan adalah kemampuan untuk menurunkan sebuah class ke class lain
- Dalam artian, kita bisa membuat class Parent dan class Child
- Class Child, hanya bisa punya satu class Parent, namun satu class Parent bisa punya banyak class Child
- Saat sebuah class diturunkan, maka semua field dan method yang ada di class Parent, secara otomatis akan dimiliki oleh class Child
- Untuk melakukan pewarisan, di class child, kita harus menggunakan kata kunci extends lalu diikuti dengan nama class parent nya.

Kode : Inheritance

```
class Animal {
  // methods and fields
}

// use of extends keyword
// to perform inheritance
class Dog extends Animal {

  // methods and fields of Animal
  // methods and fields of Dog
}
```

- Pada contoh di atas, dapat dilihat bahwa classs Dog dibuat dengan menurunkan sifat dan methods dari class Animal
- Sehingga dapat dikatakan Class Dog merupakan subclass dari Class Animal dan Animal merupakan superclass

Contoh Java Inheritance

UNIVERSITAS TEKNOLOGI SUMBAWA

```
class Animal {

  // field and method of the parent class
  String name;
  public void eat() {
    System.out.println("I can eat");
  }
}

// inherit from Animal
class Dog extends Animal {

  // new method in subclass
  public void display() {
    System.out.println("My name is " + name);
  }
}
```

Contoh Java Inheritance

UNIVERSITAS TEKNOLOGI SUMBAWA

```
class Main {
  public static void main(String[] args) {

    // create an object of the subclass
    Dog labrador = new Dog();

    // access field of superclass
    labrador.name = "Rohu";
    labrador.display();

    // call method of superclass
    // using object of subclass
    labrador.eat();

  }
}
```

Contoh Java Inheritance

UNIVERSITAS TEKNOLOGI SUMBAWA

- Output:

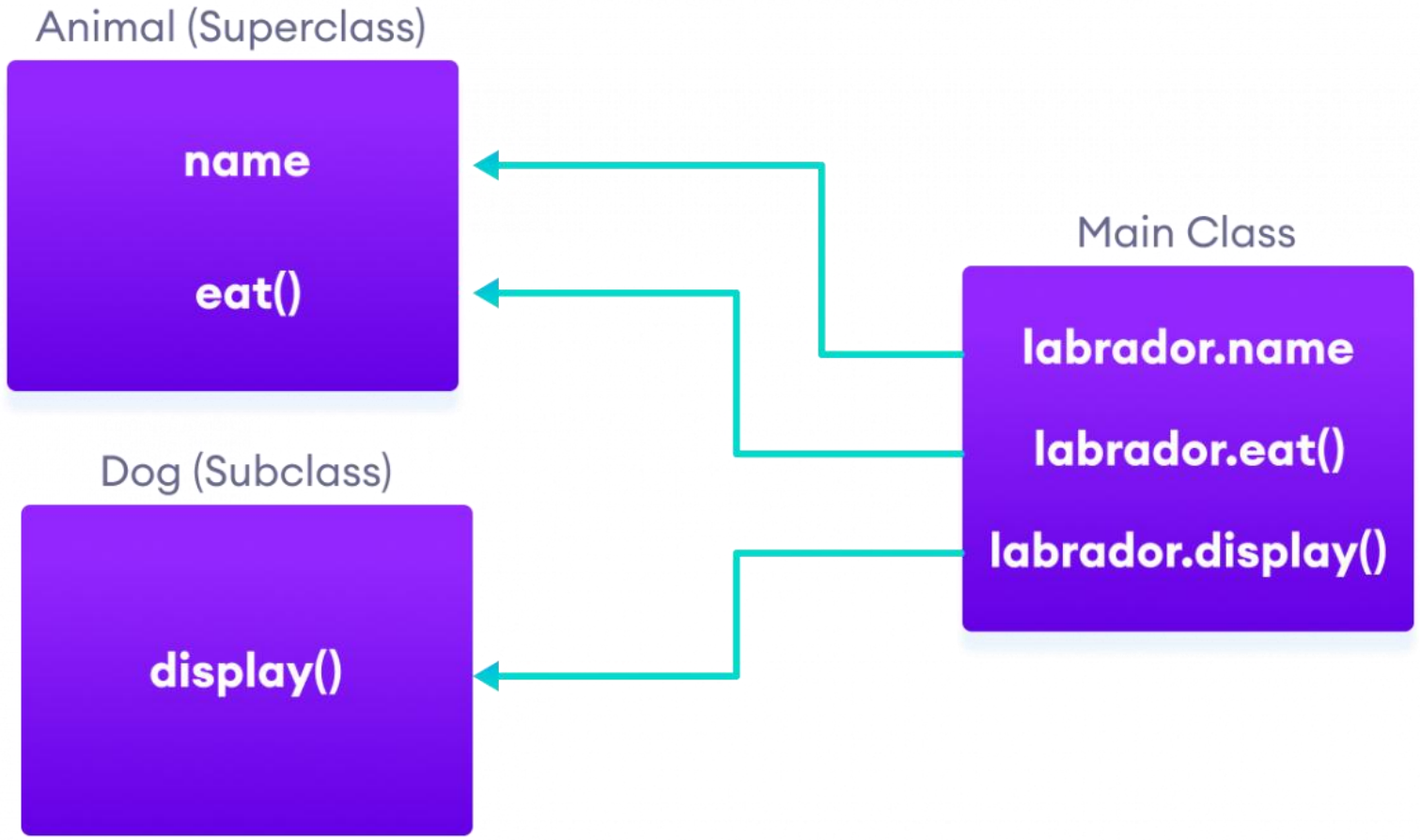
```
My name is Rohu
I can eat
```

- Dalam contoh di atas, kita telah menurunkan subkelas Dog dari super kelas Animal Perhatikan pernyataan berikut:

```
labrador.name = "Rohu";

labrador.eat();
```

- Di dini, anjing Labrador adalah sebuah objek dari class Dog. Namun, name() dan eat() merupakan anggota dari class Hewan.
- Sejak class Dog mewarisi field dan methods dari class Animal, maka kita bisa mengakses field dan method menggunakan objek dari class Dog

Inheritance



Hubungan adalah-suatu (is-a relationship)

UNIVERSITAS TEKNOLOGI SUMBAWA

- Dalam Java, pewarisan adalah hubungan **adalah-suatu**. Artinya, kita menggunakan pewarisan hanya jika ada hubungan **adalah-suatu** antara dua kelas. Misalnya,
  - **Mobil** adalah **Kendaraan**
  - **Jeruk** adalah **Buah**
  - **Dokter Bedah** Adalah Seorang **Dokter**
  - **Anjing** adalah **Hewan**

- Di sini, **Mobil** mewarisi sifat dari **Kendaraan** , **Jeruk** mewarisi sifat dari **Buah** , dan seterusnya.

UNIVERSITAS TEKNOLOGI SUMBAWA

Method Overriding

Method Overriding

UNIVERSITAS TEKNOLOGI SUMBAWA

- Method overriding adalah kemampuan mendeklarasikan ulang method di child class, yang sudah ada di parent class
- Saat kita melakukan proses overriding tersebut, secara otomatis ketika kita membuat object dari class child, method yang di class parent tidak bisa diakses lagi

Kode : Method Overriding

UNIVERSITAS TEKNOLOGI SUMBAWA

```
class Animal {

  // method in the superclass
  public void eat() {
    System.out.println("I can eat");
  }
}

// Dog inherits Animal
class Dog extends Animal {

  // overriding the eat() method
  @Override
  public void eat() {
    System.out.println("I eat dog food");
  }

  // new method in subclass
  public void bark() {
    System.out.println("I can bark");
  }
}
```

Kode : Method Overriding

UNIVERSITAS TEKNOLOGI SUMBAWA

```
class Main {
  public static void main(String[] args) {

    // create an object of the subclass
    Dog labrador = new Dog();

    // call the eat() method
    labrador.eat();
    labrador.bark();
  }
}
```

Kode : Mengakses Method Overriding

UNIVERSITAS TEKNOLOGI SUMBAWA

- Output

```
I eat dog food
I can bark
```

- Dalam contoh di atas, method eat() ini hadir di super class Animal dan sub class Dog.
- Di sini, kita telah membuat sebuah objek Labrador dari class Dog.
- Sekarang ketika kita memanggil method eat() menggunakan objek Labrador, maka compiler akan memanggil method eat() di dalam class Dog. Hal ini karena method di dalam class turunan menggantikan method di dalam class dasar, Ini disebut Overriding.

UNIVERSITAS TEKNOLOGI SUMBAWA

Super Keyword

Super Keyword

UNIVERSITAS TEKNOLOGI SUMBAWA

- Kadang kita ingin mengakses method yang terdapat di class parent yang sudah terlanjur kita override di class child
- Untuk mengakses method milik class parent, kita bisa menggunakan kata kunci super
- Sederhananya, super digunakan untuk mengakses class parent
- Tidak hanya method, field milik parent class pun bisa kita akses menggunakan kata kunci super

Kode : Super Keyword

UNIVERSITAS TEKNOLOGI SUMBAWA

```
class Animal {

  // method in the superclass
  public void eat() {
    System.out.println("I can eat");
  }
}

// Dog inherits Animal
class Dog extends Animal {

  // overriding the eat() method
  @Override
  public void eat() {

    // call method of superclass
    super.eat();
    System.out.println("I eat dog food");
  }

  // new method in subclass
  public void bark() {
    System.out.println("I can bark");
  }
}

class Main {
  public static void main(String[] args) {
```

```
class Main {
  public static void main(String[] args) {

    // create an object of the subclass
    Dog labrador = new Dog();

    // call the eat() method
    labrador.eat();
    labrador.bark();
  }
}
```

Output:

```
aku bisa makan
Saya makan makanan anjing
aku bisa menggonggong
```

Kode : Super Keyword

UNIVERSITAS TEKNOLOGI SUMBAWA

- Dalam contoh di atas, methods eat() ini hadir di kelas dasar Animal dan kelas turunan Dog. Perhatikan pernyataan berikut:

```
super.eat();
```

- Di sini, kata kunci **super** digunakan untuk memanggil method eat() yang ada di superkelas.
- Kita juga dapat menggunakan kata kunci super untuk memanggil konstruktor superclass dari konstruktor subclass.

UNIVERSITAS TEKNOLOGI SUMBAWA

Super Constructor

Super Constructor

UNIVERSITAS TEKNOLOGI SUMBAWA

- Tidak hanya untuk mengakses method atau field yang ada di parent class, kata kunci super juga bisa digunakan untuk mengakses constructor
- Namun syaratnya untuk mengakses parent class constructor, kita harus mengaksesnya di dalam class child constructor
- Jika sebuah class parent tidak memiliki constructor yang tidak ada parameter-nya (tidak memiliki default constructor), maka class child wajib mengakses constructor class parent tersebut.

Kode : Super Constructor

UNIVERSITAS TEKNOLOGI SUMBAWA

```
    Manager(String name){
        this.name = name;
    }

}

class VicePresident extends Manager {

    VicePresident(String name){
        super(name);
    }
```

Kode : Menggunakan Super Constructor

UNIVERSITAS TEKNOLOGI SUMBAWA

```
    var manager = new Manager( name: "Eko");
    manager.sayHello( name: "Budi");

    var vicePresident = new VicePresident( name: "Kurniawan");
    vicePresident.sayHello( name: "Budi");

}
}
```

UNIVERSITAS TEKNOLOGI SUMBAWA

Protected Members

Protected Members

UNIVERSITAS TEKNOLOGI SUMBAWA

- Protected Member di Java berarti Method tersebut hanya bisa di akses dari sub class atau class turunan dari class tersebut

Contoh Protected Members

```
class Animal {
  protected String name;

  protected void display() {
    System.out.println("I am an animal.");
  }
}

class Dog extends Animal {

  public void getInfo() {
    System.out.println("My name is " + name);
  }
}

class Main {
  public static void main(String[] args) {

    // create an object of the subclass
    Dog labrador = new Dog();

    // access protected field and method
    // using the object of subclass
    labrador.name = "Rocky";
    labrador.display();

    labrador.getInfo();
  }
}
```

- Output

```
I am an animal.
My name is Rocky
```

- Pada contoh di samping, kita telah membuat sebuah Class Animal. Class Animal memiliki Protected Field: name dan sebuah methods display()
- Kita memiliki class Dog yang merupakan turunan dari class Animal. Perhatikan kode berikut:

```
labrador.name = "Rocky";
labrador.display();
```

- Dapat kita lihat, kita bisa mengakses method yang ada pada superclass melalui object Labarador

Mengapa menggunakan Inheritance?

UNIVERSITAS TEKNOLOGI SUMBAWA

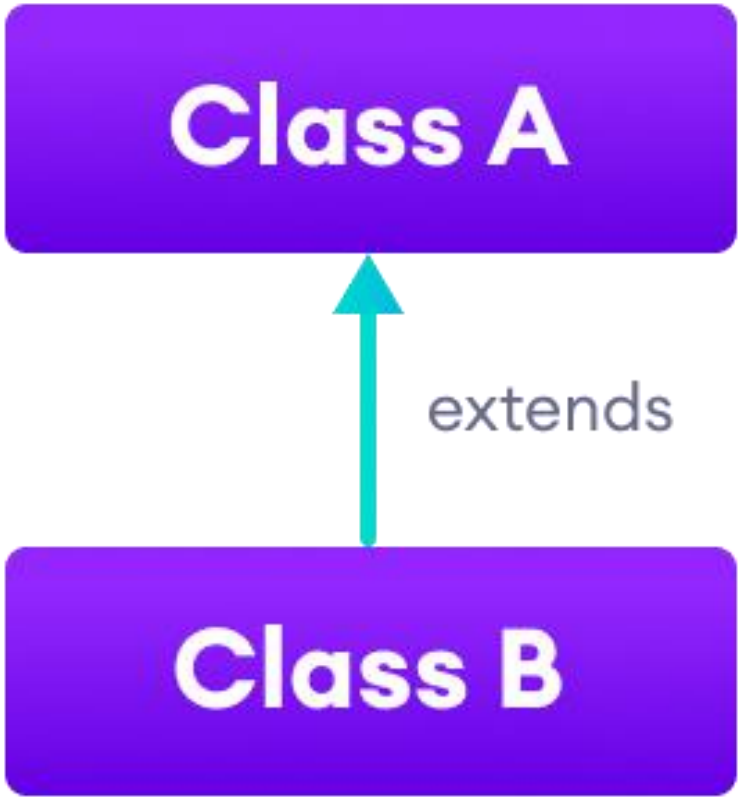
- Penggunaan Inheritance yang paling penting di Java adalah penggunaan kembali kode. Kode yang ada di kelas induk dapat langsung digunakan oleh kelas anak.
- Overriding metods juga dikenal sebagai Polimorfisme saat dijalankan. Oleh karena itu, kita dapat mencapai Polimorfisme di Java dengan Inheritance.

UNIVERSITAS TEKNOLOGI SUMBAWA

Types of inheritance

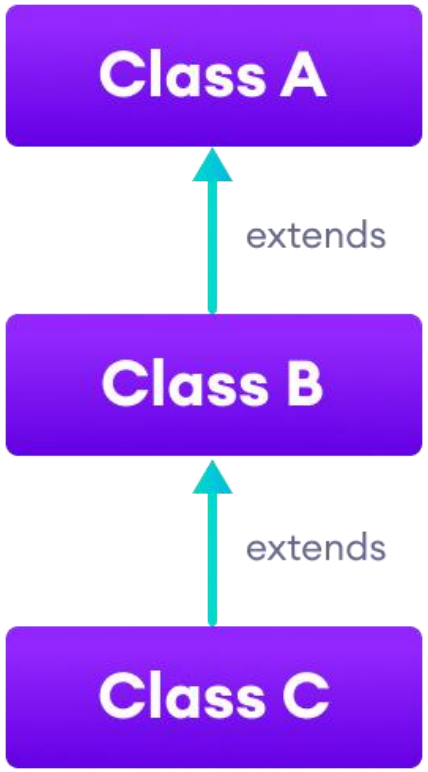
1. Single Inheritance

- In single inheritance, a single subclass extends from a single superclass. For example,



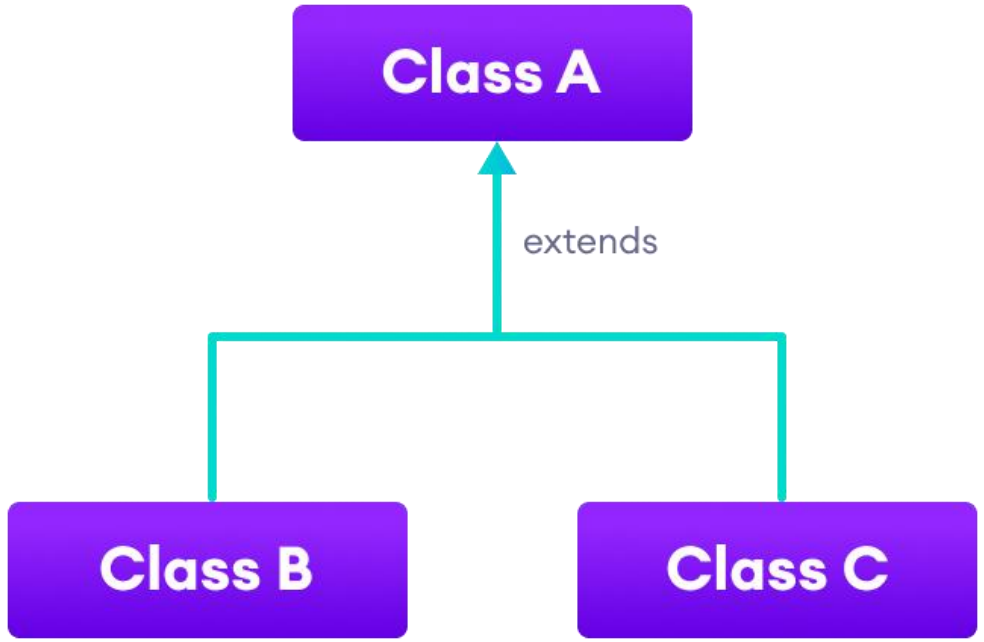
2. Multilevel Inheritance

- In multilevel inheritance, a subclass extends from a superclass and then the same subclass acts as a superclass for another class. For example



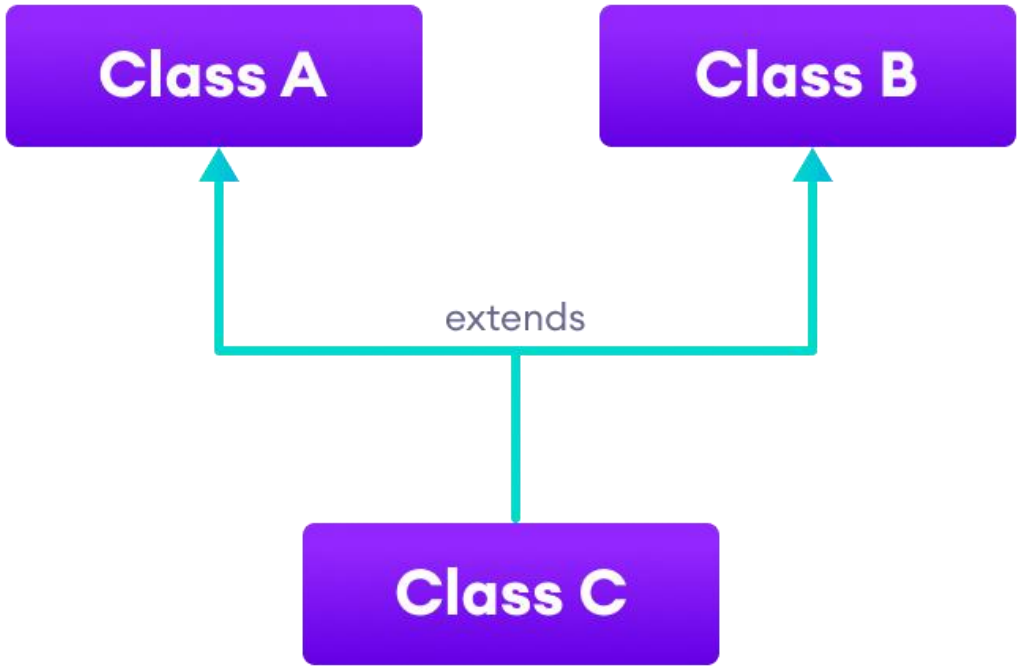
3. Hierarchical Inheritance

- In hierarchical inheritance, multiple subclasses extend from a single superclass. For example,



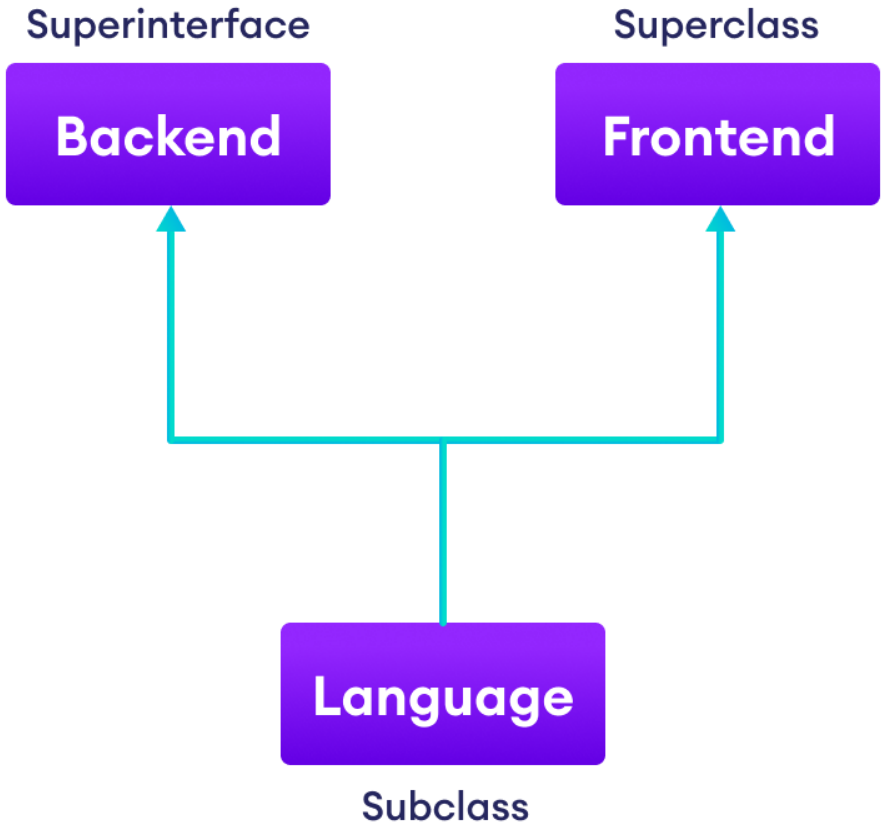
4. Multiple Inheritance

- In multiple inheritance, a single subclass extends from multiple superclasses. For example,



Multiple Inheritance in Java

- Multiple Inheritance adalah kondisi dimana sebuah Child Class mewarisi sifat lebih dari satu super class.
- Pada dasarnya Java tidak mendukung Multiple Inheritance, untuk dapat menggunakan Multiple Inheritance di Java, kita harus menggunakan interface.



Multiple Inheritance in Java

UNIVERSITAS TEKNOLOGI SUMBAWA

```
interface Backend {

  // abstract class
  public void connectServer();
}

class Frontend {

  public void responsive(String str) {
    System.out.println(str + " can also be used as frontend.");
  }
}
```

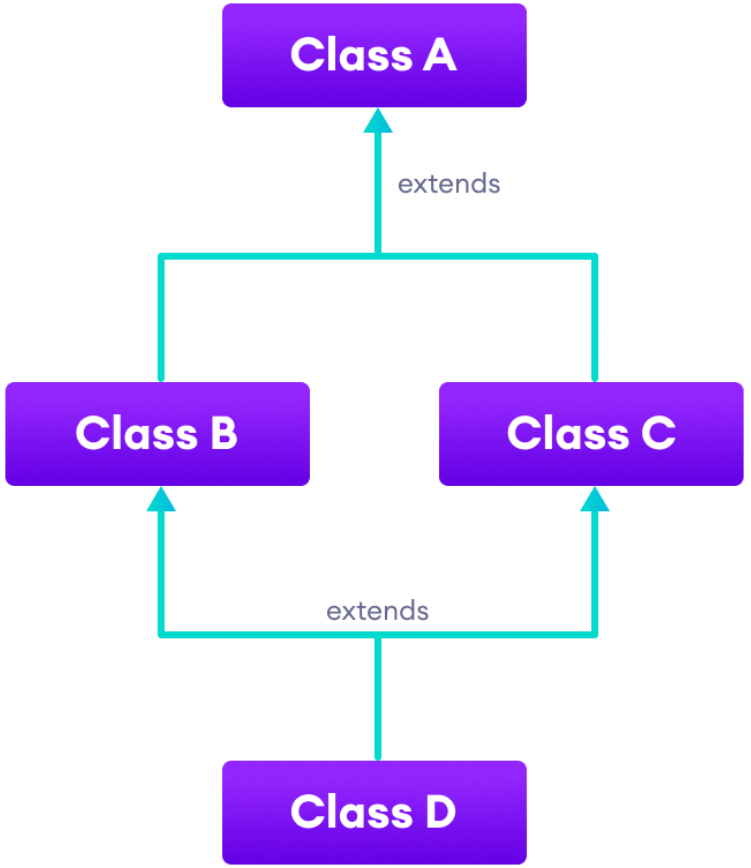
```
// Language extends Frontend class
// Language implements Backend interface
class Language extends Frontend implements Backend {

  String language = "Java";

  // implement method of interface
  public void connectServer() {
    System.out.println(language + " can be used as backend language.");
  }

  public static void main(String[] args) {

    // create object of Language class
    Language java = new Language();

    java.connectServer();

    // call the inherited method of Frontend class
    java.responsive(java.language);
  }

}
```

Output :

```
Java can be used as backend language.
Java can also be used as frontend.
```

5. Hybrid Inheritance

- Hybrid inheritance is a combination of two or more types of inheritance. For example,



UNIVERSITAS TEKNOLOGI SUMBAWA

Object Class

Object Class

UNIVERSITAS TEKNOLOGI SUMBAWA

- Di Java, setiap class yang kita buat secara otomatis adalah turunan dari class Object
- Walaupun tidak secara langsung kita eksplisit menyebutkan extends Object, tapi secara otomatis Java akan membuat class kita extends Object
- Bisa dikatakan class Object adalah superclass untuk semua class yang ada di Java

Isi Class Object

Kode : Menggunakan Class Object

UNIVERSITAS TEKNOLOGI SUMBAWA

```
    var manager = new Manager( name: "Eko");
    System.out.println(manager); // otomatis memanggil method toString()
    System.out.println(manager.toString());

  }
}
```

UNIVERSITAS TEKNOLOGI SUMBAWA

Terima kasih