

# Pemrograman Berorientasi Obyek

## Inheritance

Oleh Politeknik Elektronika Negeri Surabaya  
2020




Politeknik Elektronika Negeri Surabaya  
Departemen Teknik Informatika dan Komputer

1

Politeknik Elektronika Negeri Surabaya

## Inheritance

- Inheritance (Pewarisan) merupakan salah satu dari tiga konsep dasar OOP.
- Inheritance bisa disebut sebagai **Generalisasi/Generalization**
- Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan.
- Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.



Departemen Teknik Informatika & Komputer

2

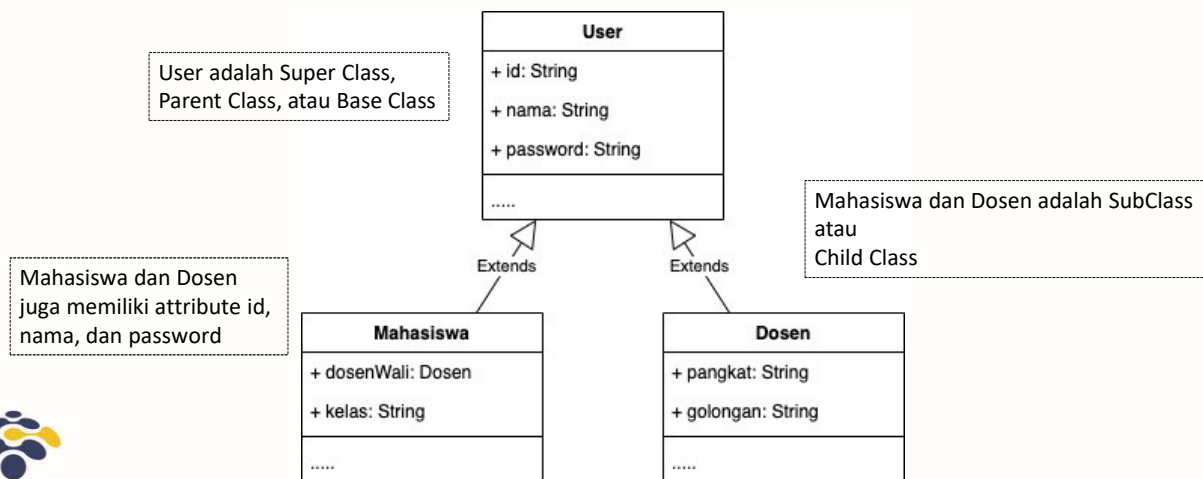
# Inheritance

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



3

# Inheritance dalam Class Diagram



4

## Pengertian Inheritance

- Karena suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya.
- Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.



## Deklarasi Inheritance

- Dengan menambahkan kata kunci **extends** setelah deklarasi nama class, kemudian diikuti dengan nama parent class-nya.
- Kata kunci extends tersebut memberitahu kompiler Java bahwa kita ingin melakukan perluasan class.



## Deklarasi Inheritance

```
public class A {
```

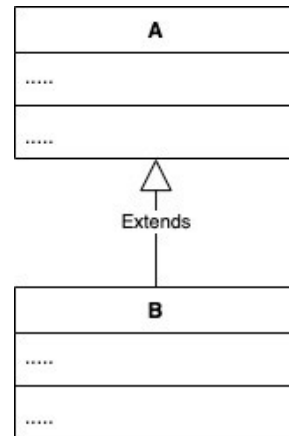
```
    ...
```

```
}
```

```
public class B extends A {
```

```
    ...
```

```
}
```



- Semua class di dalam Java adalah merupakan subclass dari class super induk yang bernama **Object**.
- Misalnya saja terdapat sebuah class sederhana :

```
public class A {
```

```
    ...
```

```
}
```



- Pada saat dikompilasi, Kompiler Java akan membacanya sebagai subclass dari class Object.

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

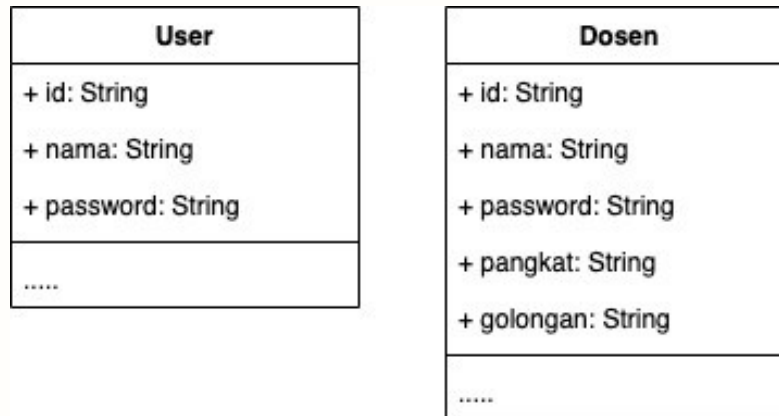


## Kapan kita menerapkan inheritance?

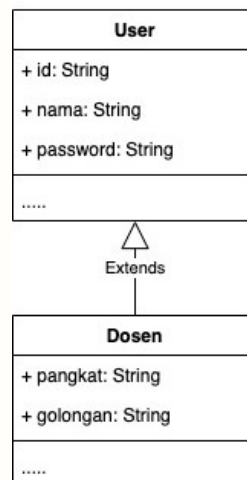
- Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain.
- Atau ketika ada beberapa kelas yang memiliki kesamaan, maka kita bisa melakukan **generalisasi** beberapa class tersebut menjadi sebuah parent class.



## Class memiliki kesamaan



## Class Dosen cukup meng-extends User



## Implementasi Kode Program

```
public class User {
    public String id;
    public String nama;
    public String password;
}
```

```
public class Dosen extends User{
    public String pangkat;
    public String golongan;
}
```

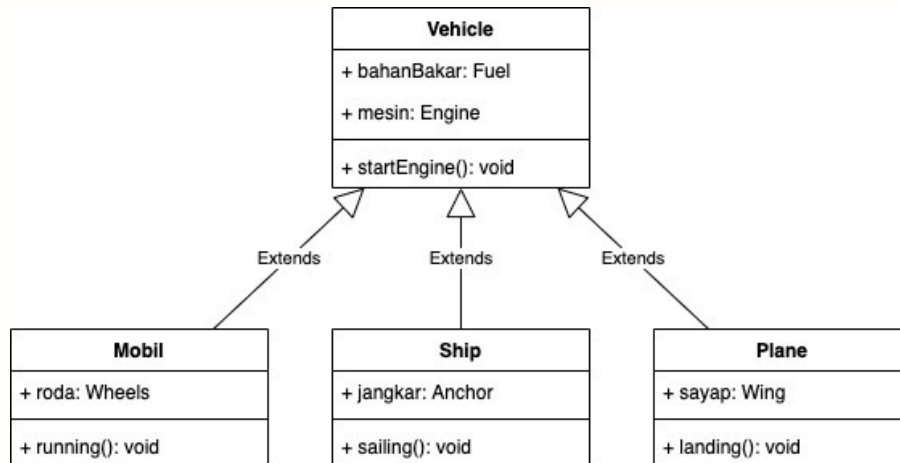


## Ketika beberapa class memiliki kesamaan

Mobil	Ship	Plane
+ bahanBakar: Fuel	+ bahanBakar: Fuel	+ bahanBakar: Fuel
+ mesin: Engine	+ mesin: Engine	+ mesin: Engine
+ roda: Wheels	+ jangkar: Anchor	+ sayap: Wing
+ startEngine(): void	+ startEngine(): void	+ startEngine(): void
+ running(): void	+ sailing(): void	+ landing(): void



## Generalisasi



## Single Inheritance

- Konsep inheritance yang ada di Java adalah Java hanya memperkenankan adanya **single inheritance**.
- Konsep single inheritance hanya memperbolehkan suatu subclass mempunyai satu parent class.





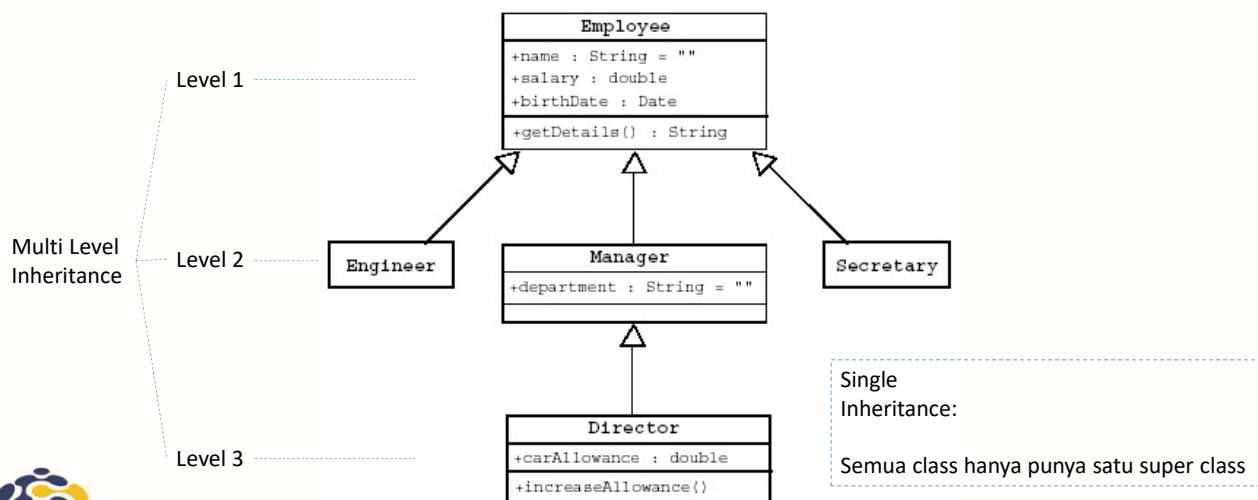
## Multilevel Inheritance

- Konsep inheritance yang ada di Java memperkenankan adanya **multilevel inheritance**.
- Konsep multilevel inheritance memperbolehkan suatu subclass mempunyai subclass lagi.



17

## Single dan Multilevel Inheritance



18

## Pengaksesan Member Super Class

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

- Pengaksesan member yang ada di parent class dari subclass-nya tidak berbeda dengan pengaksesan member subclass itu sendiri.



```
public class Manajer extends Pegawai {
    public String departemen;

    public void IsiData(String n, String d) {
        nama=n;
        departemen=d;
    }
}
```

## Kata kunci super

- Kata kunci super dipakai untuk merujuk pada member dari parent class.
- Sebagaimana kata kunci this yang dipakai untuk merujuk pada member dari class itu sendiri.
- Format penulisannya adalah sebagai berikut :
  - super.data\_member  
→ merujuk pada data member pada parent class
  - super.function\_member()  
→ merujuk pada function member pada parent class
  - super()  
→ merujuk pada konstruktor pada parent class



## Contoh

```
class Parent {
    public int x = 5;
}

class Child extends Parent {
    public int x = 10;

    public void info(int x) {
        System.out.println("Nilai x sebagai parameter = " + x);
        System.out.println("Data member x di class Child = " + this.x);
        System.out.println("Data member x di class Parent = " +
            super.x);
    }
}
```

```
public class NilaiX {
    public static void main(String args[]) {
        Child tes = new Child();
        tes.info(20);
    }
}
```

### Hasil Running

- Nilai x sebagai parameter = 20
- Data member x di class Child = 10
- Data member x di class Parent = 5



## Kesimpulan

- **x**  
→ merujuk pada x terdekat, yaitu parameter Info()
- **this.x**  
→ merujuk pada data member dari class-nya sendiri, yaitu data member pada class Child
- **super.x**  
→ merujuk pada data member dari parent class-nya, yaitu data member pada class Parent



## Konstruktor tidak diwariskan

- Konstruktor dari parent class tidak dapat diwariskan ke subclass-nya.
- Konsekuensinya, setiap kali kita membuat suatu subclass, maka kita harus memanggil konstruktor parent class di konstruktor subclass.
- Pemanggilan konstruktor parent harus dilakukan pada baris pertama dari konstruktor subclass.
- Jika kita tidak mendeklarasikannya secara eksplisit, maka kompiler Java akan menambahkan deklarasi pemanggilan konstruktor parent class di konstruktor subclass.



## Konstruktor tidak diwariskan

- Sebelum subclass menjalankan konstruktornya sendiri, subclass akan menjalankan constructor superclass terlebih dahulu.
- Hal ini terjadi karena secara implisit pada constructor subclass ditambahkan pemanggilan `super ()` yang bertujuan memanggil constructor superclass oleh kompiler.



## Konstruktor tidak diwariskan

Misalnya saja kita mempunyai dua buah class sebagai berikut :

```
public class Parent  
{  
}
```

```
public class Child extends Parent {  
}
```



- Pada saat program tersebut dikompilasi, maka kompiler Java akan menambahkan :
  - konstruktor class Parent
  - konstruktor class Child
  - pemanggilan konstruktor class Parent di kostruktor class Child





## Rangkuman

- Inheritance adalah pewarisan
- Child mewarisi parent
- Yang diwarisi adalah attribute dan method
- Tidak semua attribute dan method dapat diwariskan
- Hanya method dan attribute yang memiliki modifier public dan protected yang diwariskan.



**bridge to the future**

<http://www.eepis-its.edu>

## Tugas

1. Apa yang dimaksud dengan inheritance?
2. Buatlah contoh kasus yang menerapkan konsep inheritance !
3. Adakah perbedaan cara mengakses member class milik parent dan member class milik sendiri? Jelaskan melalui contoh ! (Silahkan memanfaatkan jawaban soal nomor 2.)
4. Apa yang dimaksud dengan konsep single inheritance ?
5. Apa yang dimaksud dengan konsep multi level inheritance ?
6. Ada berapa modifier untuk pengontrolan akses? Jelaskan masing-masing!
7. Apakah kegunaan kata kunci super? Jelaskan !
8. Apakah yang dimaksud dengan konstruktor tidak diwariskan?



31

1. Oracle Java Documentation, The Java™ Tutorials, <https://docs.oracle.com/javase/tutorial/>, Copyright © 1995, Oracle 2015.
2. Tita Karlita, Yuliana Setrowati, Rizky Yuniar Hakkun, Pemrograman Berorientasi Obyek, PENS-2012
3. Sun Java Programming, Sun Educational Services, Student Guide, Sun Microsystems, 2001. **bridge to the future**
4. John R. Hubbard, Programming With Java, McGraw-Hill, ISBN: 0-07-142040-1, 2004.
5. Patrick Niemeyer, Jonathan Knudsen, Learning Java, O'reilly, CA, ISBN: 1565927184, 2000.
6. Philip Heller, Simon Roberts, Complete Java 2 Certification Study Guide, Third Edition, Sybex, San Francisco, London, ISBN: 0-7821-4419-5, 2002.
7. Herbert Schildt, The Complete Reference, Java™ Seventh Edition, Mc Graw Hill, Osborne, ISBN: 978-0-07-163177-8, 2007

32