PEMROGRAMAN BERORIENTASI OBJEK

INHERITANCE

Faisal Fajri Rahani, S.Si., M.Cs.

Informatika

2022

Inheritance

- Inheritance
- Multiple Inheritance

Inheritance

- **Inheritance** atau **Pewarisan/Penurunan** adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ property dan method yang dimilikinya kepada class lain.
- Konsep inheritance digunakan untuk memanfaatkan fitur **‘code reuse’** untuk menghindari duplikasi kode program. (Sumber)
- Konsep inheritance ini mengadopsi dunia Konsep inheritance ini mengadopsi dunia riil dimana suatu entitas/obyek dapat mempunyai entitas/obyek turunan.
- Dengan konsep inheritance, sebuah class dapat mempunyai class turunan.

Pengertian dasar inheritance

- Suatu class yang mempunyai class 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.
- Karena suatu subclass dapat mewarisi apa 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 (extend) parent class-nya.

Deklarasi inheritance

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

Deklarasi inheritance

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

Semua class di dalam Java adalah merupakan sub class dari class super induk yang bernama **Object**.

- Misalnya saja terdapat sebuah class sederhana :

```
public class A {
        …
}
```

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

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

Kapan kita menerapkan inheritance?

- Kita baru perlu menerapkan inheritance Kita baru perlu menerapkan inheritance pada saat kita jumpai ada suatu class yang dapat diperluas dari class lain yang dapat diperluas dari class lain.

Keuntungan Inheritance

- Keuntungan dari Inheritance adalah Reusability
  - Tidak harus menyalin semua data dan method dari suatu kelas jika akan menggunakannya lagi
- Kemudahan dalam me-manage kelas yang memiliki data dan method yang sama
  - Sekali perilaku(method) didefinisikan pada super class, maka perilaku perilaku tersebut tersebut secara otomatis otomatis diwariskan ke subclass. Sehingga hanya perlu menulis method sekali dan bisa digunakan untuk semua subclass.
  - Sekali properti/field di definisikan di superclass, maka semua properti di wariskan ke semua subclass. Superclass dan subclass berbagi properti
  - Subclass hanya perlu mengimplementasikan jika ada perbedaan dengan parentnya.

Contoh

Contoh terdapat class Pegawai

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

Contoh terdapat class Manager

```
public class Manajer {
        public String nama;
        public double gaji;
        public String departemen;
}
```

- Dari 2 buah class diatas, kita lihat class Manajer mempunyai data member yang identik sama dengan class Pegawai, hanya saja ada tambahan data member departemen.
- Sebenarnya yang terjadi disana adalah class Manajer merupakan perluasan dari class Pegawai dengan tambahan data member departemen tambahan data member departemen.
- Disini perlu memakai konsep inheritance, sehingga class Manajer dapat kita tuliskan seperti berikut

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

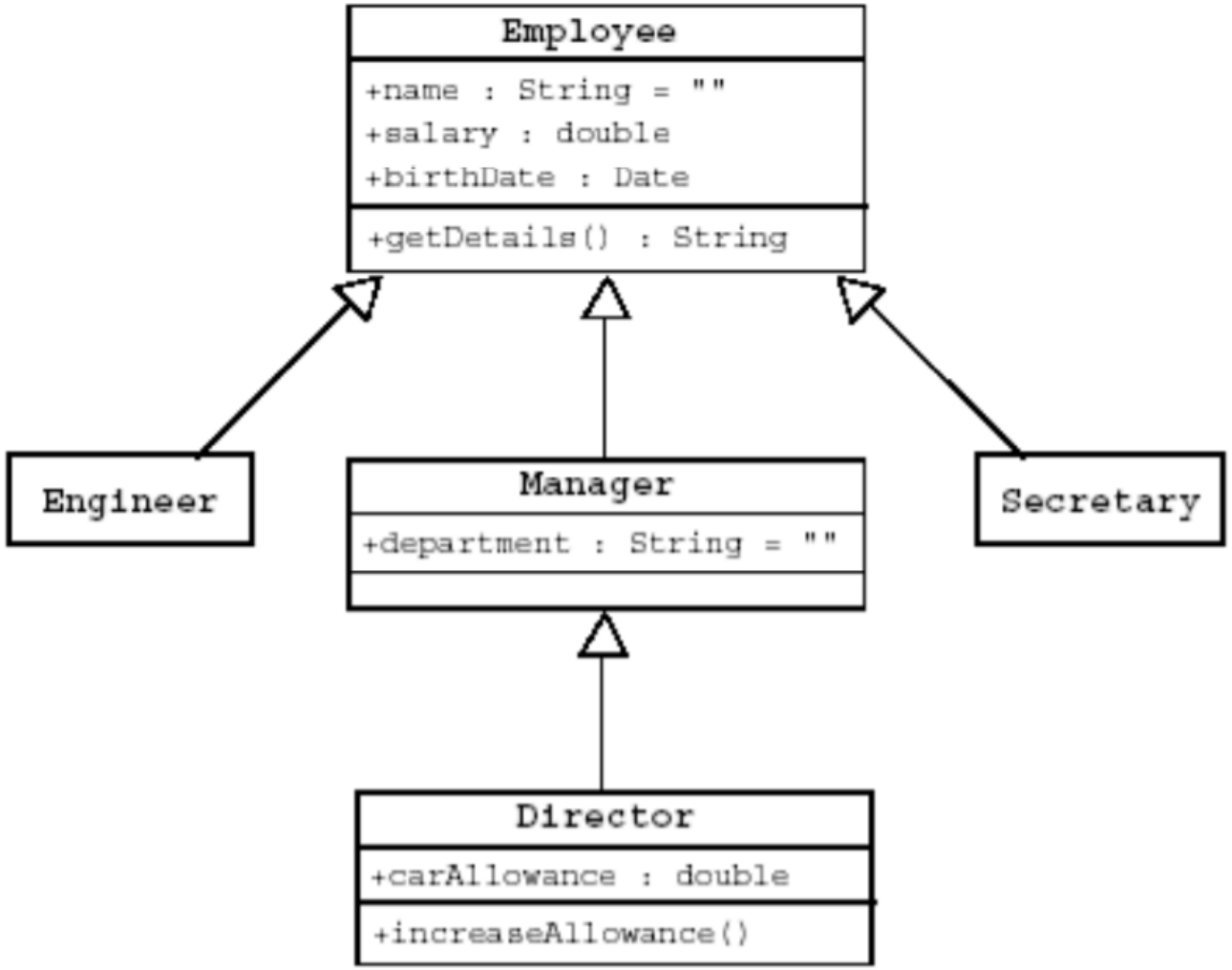
Single Inheritance

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

Multilevel Inheritance

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

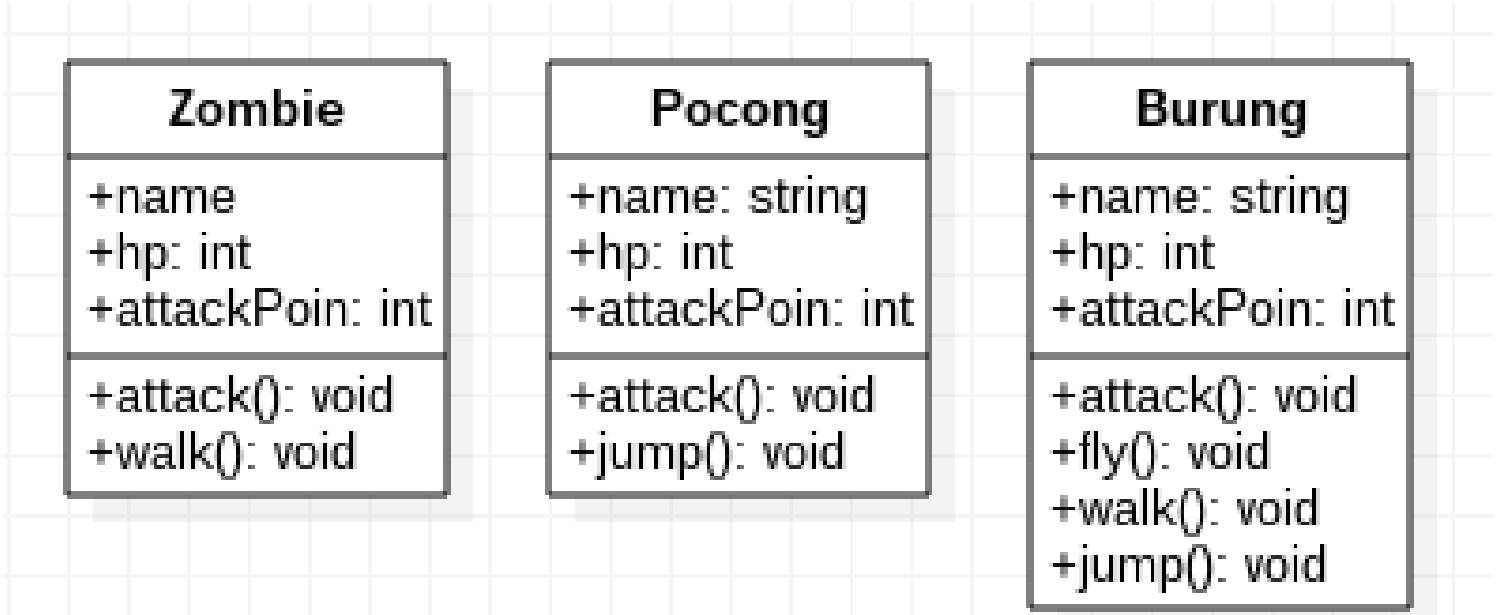
Single dan Multilevel Inheritance



Contoh tanpa Inheritance

Sumber

- Misalkan dalam Game, kita akan membuat class-class musuh dengan perilaku yang berbeda.



Lalu kita membuat kode untuk masing-masing kelas seperti ini:

Zombie.java

```
class Zombie {
    String name;
    int hp;
    int attackPoin;

    void attack(){
        // ...
    }

    void walk(){
        //...
    }
}
```

Pocong.java

```
class Pocong {
    String name;
    int hp;
    int attackPoin;

    void attack(){
        // ...
    }

    void jump(){
        //...
    }
}
```

Burung.java

```
class Burung {
    String name;
    int hp;
    int attackPoin;

    void attack(){
        // ...
    }

    void walk(){
        //...
    }

    void jump(){
        //...
    }

    void fly(){
        //...
    }
}
```

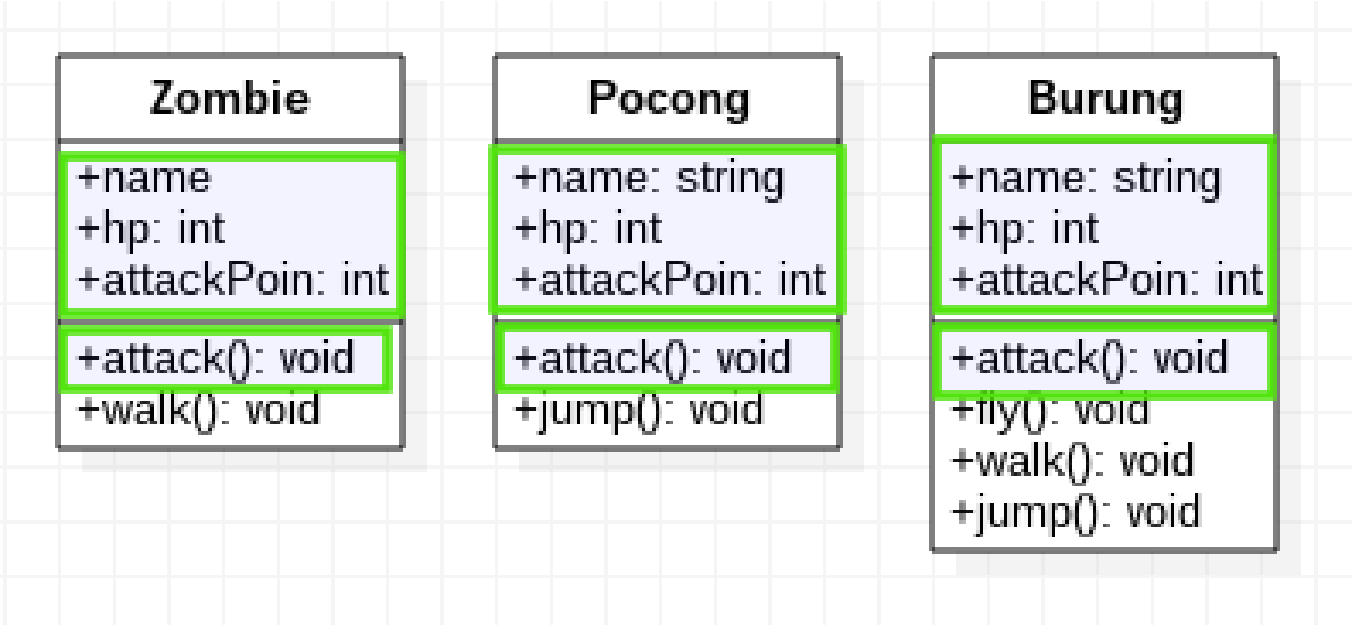
- Apakah boleh seperti ini?

- Ya, boleh-boleh saja. Akan Tapi tidak efektif, karena kita menulis berulang-ulang properti dan method yang sama.

Contoh dengan Inheritance

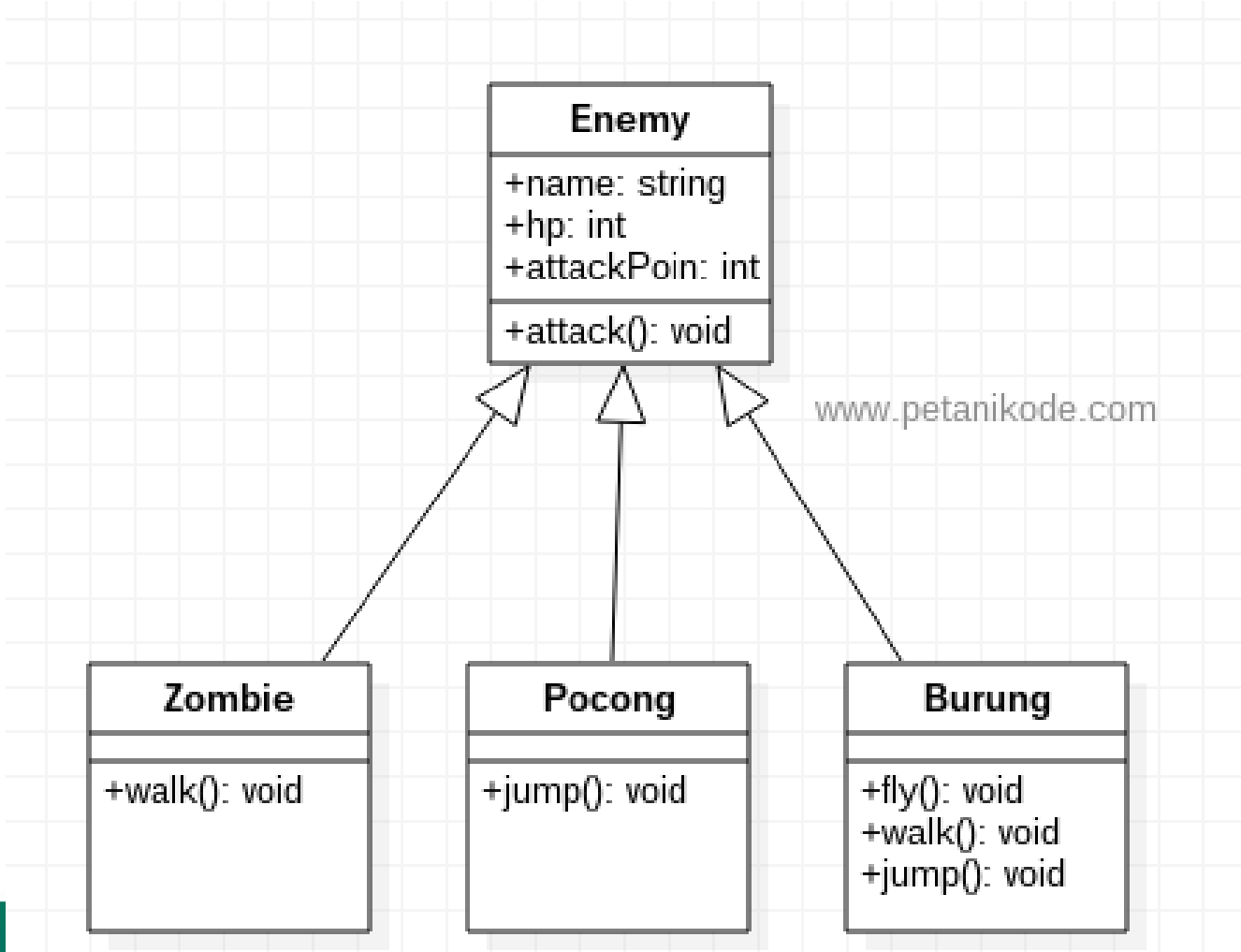
Mari kita lihat memeber class yang sama

- Ada hal yang sama pada class dibawah



Menggunakan inheritance

- Inheritance di StarUML digambarkan dengan garis hubung Generalization.
- Class Enemy adalah class induk yang memiliki anak Zombie, Pocong, dan Burung. Apapun properti yang ada di class induk, akan dimiliki juga oleh class anak.



Sumber

Hasil kode

Enemy.java

```
class Enemy {
    String name;
    int hp;
    int attackPoin;

    void attack(){
        System.out.println("Serang!");
    }
}
```

Zombie.java

```
class Zombie extends Enemy {
    void walk(){
        System.out.println("Zombie jalan-jalan");
    }
}
```

- Pada class anak, kita menggunakan kata kunci extends untuk menyatakan kalau dia adalah class turunan dari Enemy.

Pocong.java

```
class Pocong extends Enemy {
    void jump(){
        System.out.println("loncat-loncat!");
    }
}
```

- Pada class anak, kita menggunakan kata kunci extends untuk menyatakan kalau dia adalah class turunan dari Enemy.

Burung.java

```
class Burung extends Enemy {
    void walk(){
        System.out.println("Burung berjalan");
    }
    void jump(){
        System.out.println("Burung loncat-loncat");
    }
    void fly(){
        System.out.println("Burung Terbang...");
    }
}
```

- Pada class anak, kita menggunakan kata kunci extends untuk menyatakan kalau dia adalah class turunan dari Enemy.

- Lalu, bila kita ingin membuat objek dari class-class tersebut, Kita bisa membuatnya seperti ini:

```
Enemy monster = new Enemy();
Zombie zumbi = new Zombie();
Pocong hantuPocong = new Pocong();
Burung garuda = new Burung();
```

Terima Kasih