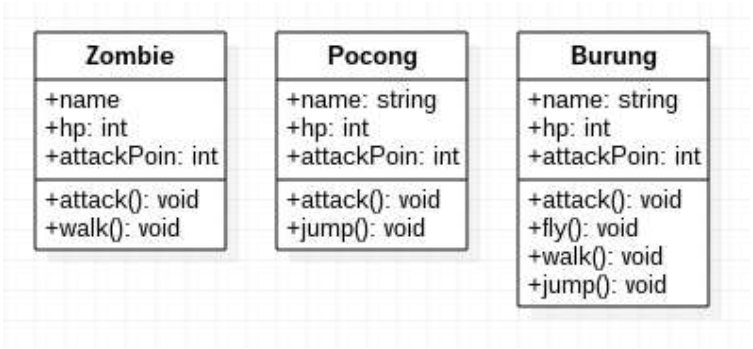
BAB V
INHERITANCE

Sebuah class atau objek bisa saling berhubungan dengan class yang lain. Salah satu bentuk hubungan tersebut adalah *inheritance* (pewarisan). Hubungan ini seperti hubungan keluarga antara orang tua dan anak. Sebuah class di Java, bisa memiliki satu atau lebih keturunan atau class anak. Class anak akan memiliki warisan properti dan method dari class ibu.

Kenapa Kita Harus Menggunakan Inheritance?

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



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

File: Zombie.java

```
class Zombie {
   String name;
   int hp;
   int attackPoin;
   void attack(){
       // ...
   }
```

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

File: Pocong.java

```
class Pocong {
    String name;
    int hp;
    int attackPoin;
    void attack(){
        // ...
    }
    void jump(){
        //...
    }
}
```

File: Burung.java

```
class Burung {
    String name;
    int hp;
    int attackPoin;
    void attack(){
        // ...
    }
    void walk(){
        //...
    }
    void jump(){
        //...
    }
    void fly(){
```

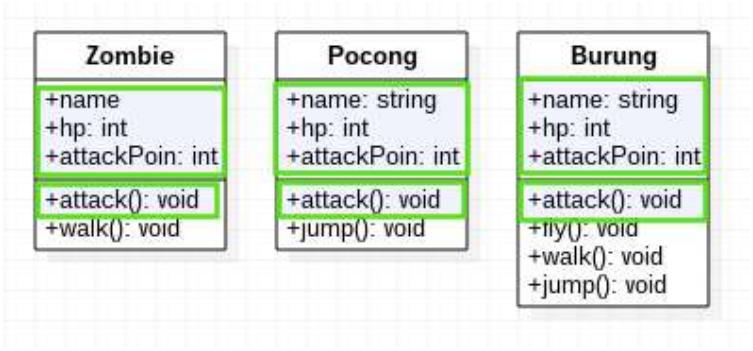
```
        //...
    }
}
```

Apakah boleh seperti ini?

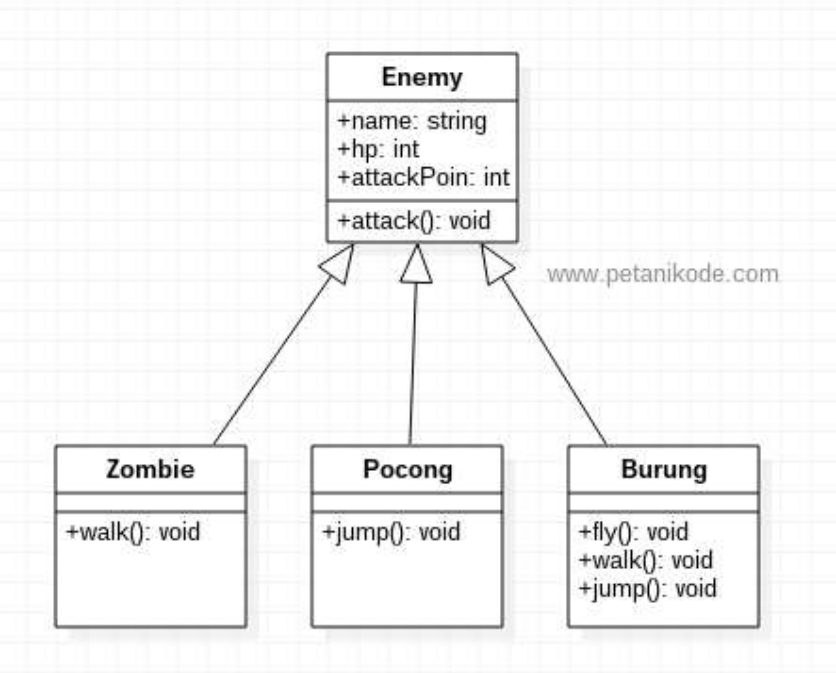
Ya, boleh-boleh saja. Akan Tapi tidak efektif, karena kita menulis berulang-ulang properti dan method yang sama. Dalam programming ini disebut sebagai WET (We Enjoy Typing) sebagai programmer kita seharusnya selalu menerapkan konsep DRY (Don't Repeat Yourself).

Maka diciptakanlah sebuah solusi yaitu *inheritance*.

Mari kita lihat member class yang sama:



Setelah menggunakan *inheritance*, maka akan menjadi seperti ini:



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.

Lalu bagaimana bentuk kodenya dalam Java?

Bentuk kodenya akan seperti ini:

File: Enemy.java

```
class Enemy {
   String name;
   int hp;
   int attackPoin;
   void attack(){
       System.out.println("Serang!");
   }
```

```
}
```

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

File: Zombie.java

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

File: Pocong.java

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

File: 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...");
   }
}
```

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

File: Main.java

```
public class Main {

   public static void main(String[] args) {
       // namaClass namaVariabel = new namaClass();
       // hati-hati case sensitive!
       Enemy monster = new Enemy();

       Zombie zumbi = new Zombie();

       Pocong hantuPocong = new Pocong();

       Burung garuda = new Burung();

       // output: Burung Terbang...
       garuda.fly();
       // output: Serang!
       garuda.attack();
       // output: Serang!
       monster.attack();
   }
}
```

"konsep pewarisan" dikelompokkan menjadi dua kategori:

- subclass (child) - kelas yang mewarisi dari kelas lain
- superclass (parents) - kelas yang diwarisi dari

Untuk mewarisi dari kelas, gunakan kata kunci extends. Pada contoh di bawah ini, kelas Car (subclass) mewarisi atribut dan metode dari kelas Vehicle (superclass):

```
class Car extends Vehicle {
```

```
    private String modelName = "Mustang";

    public static void main(String[] args) {
        Car myFastCar = new Car();
        myFastCar.honk();
        System.out.println(myFastCar.brand + " " +
myFastCar.modelName);
    }
}
```

Jenis-jenis Inheritance

1. **Single Inheritance**

   Ketika suatu kelas mewarisi kelas lain, itu dikenal sebagai single inheritance. Dalam contoh yang diberikan di bawah ini, kelas Anjing mewarisi kelas Hewan, jadi ada inheritance single.

   File: TestInheritance.java

```
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();
    }
}
```

2. **Multilevel Inheritance**

    Ketika ada rantai warisan, itu dikenal sebagai inheritance multilevel. Seperti yang Anda lihat dalam contoh yang diberikan di bawah ini, kelas BabyDog mewarisi kelas Anjing yang lagi mewarisi kelas Hewan, jadi ada warisan bertingkat.

    File: TestInheritance2.java

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

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

class BabyDog extends Dog {
    void weep() {
        System.out.println("weeping...");
    }
}

class TestInheritance2 {
    public static void main(String args[]) {
        BabyDog d = new BabyDog();
        d.weep();
```

```
        d.bark();
        d.eat();
    }
}
```

3. Hierarchical Inheritance

Ketika dua atau lebih kelas mewarisi satu kelas, itu dikenal sebagai inheritance hierarchical. Dalam contoh yang diberikan di bawah ini, kelas Anjing dan Kucing mewarisi kelas Hewan, jadi ada warisan hierarkis.

File: TestInheritance3.java

```
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 c = new Cat();

       c.meow();

       c.eat();

       // c.bark();//C.T.Error

   }
}
```

Keyword "*this*" dan "*super*"

1. ***this***
   Kata kunci ***this*** digunakan sebagai referensi dari class itu sendiri.

```
class Person {

   private String name;

   public void setName(String name){

       this.name = name;

   }

}
```

Maka ***this*** yang dimaksud pada class tersebut adalah class ***Person***.

2. ***super***

   Jika ***this*** merepresentasikan objek dari class itu sendiri, maka ***super*** akan merepresentasikan objek dari class induk.

   Perhatikan contoh ini:

   Kita mencoba membuat class **Person** yang akan menjadi class induk atau super class.

   File : Person.java

```
public class Person {

   String name = "Sasuke";

   int age = 22;

}
```

   Lalu, kita buat class turunannya (sub class), yaitu: class **Employee**.

   File : Employee.java

```
public class Employee extends Person {

   float salary = 4000f;

   String name = "Sakura";

   int age = 23;

   public void showInfo(){

       System.out.println("Name: " + super.name);
```

```
        System.out.println("Age: " + super.age);

        System.out.println("Salary: $" + salary);

    }

}
```

Berikutnya kita membuat class **Demo** untuk membuat objek dan method `main()`.

File : Demo.java

```
public class Demo {

    public static void main(String[] args) {

        Employee sakura = new Employee();

        sakura.showInfo();
    }
}
```

Sekarang coba eksekusi class **Demo**, apa hasil outputnya?

```
Name: Sasuke
Age: 22
Salary: $4000.0
```

Mengapa hasil outputnya begini?

Karena pada method showInfo() kita menggunakan kata kunci ***super*** untuk mengambil nilai dari variabel yang ada di dalam class induk *(super class)*.

```
public void showInfo(){
   System.out.println("Name: " + super.name);
   System.out.println("Age: " + super.age);
   System.out.println("Salary: $" + salary);
}
```

Bila kita mengganti `super` menjadi `this`...

```
public void showInfo(){
   System.out.println("Name: " + this.name);
   System.out.println("Age: " + this.age);
   System.out.println("Salary: $" + salary);
}
```

...maka hasil outputnya akan seperti ini.

```
Name: Sakura
Age: 23
Salary: $4000.0
```