

Pemrograman Berorientasi Objek

S1 Informatika UNS

Inheritance (pewarisan)

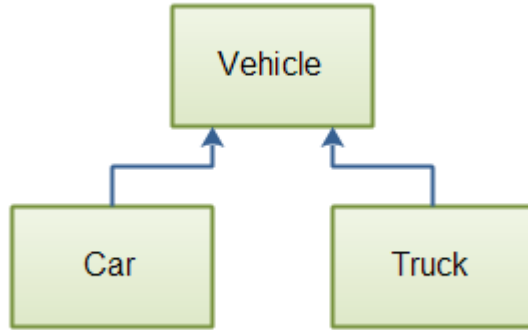
Ardhi Wijayanto, S.Kom., M.Cs.

2020

Inheritance

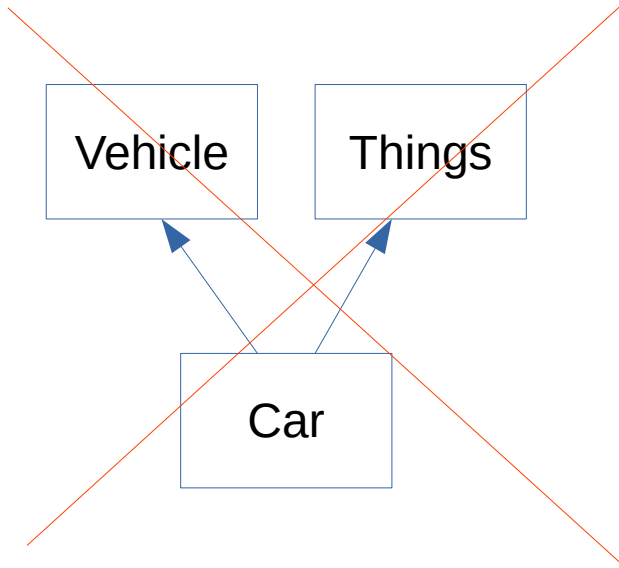
- Metode untuk code reuse
- Sebuah class diturunkan / diwariskan ke class lain
- Meniru makhluk hidup
- Class hasil turunan memiliki atribut dan method yang sama dengan class aslinya / parent

Inheritance



- Class asli / superclass = Vehicle
- Class turunan / subclass = Car dan Truck

Inheritance



- Dalam bahasa Java sebuah class hanya dapat diturunkan dari 1 class lain
- Class Car tidak bisa diturunkan / menjadi subclass dari 2 class lain
- Bahasa lain misal C++ support multiple inheritance

Apa yang diwariskan?

- Method dan variabel / atribut
 - Protected dan public modifier
- Constructor tidak diwariskan

Pewarisan dalam Source Code

```
public class Vehicle {  
    protected String licensePlate = null;  
  
    public void setLicensePlate(String license) {  
        this.licensePlate = license;  
    }  
}  
  
public class Car extends Vehicle {  
    int numberOfSeats = 0;  
  
    public int getNumberOfSeats() {  
        return this.numberOfSeats;  
    }  
}
```

Memanggil Atribut dari Superclass

```
public class Vehicle {  
    protected String licensePlate = null;  
  
    public void setLicensePlate(String license) {  
        this.licensePlate = license;  
    }  
}  
  
public class Car extends Vehicle {  
    int numberOfSeats = 0;  
  
    public int getNumberOfSeats() {  
        return this.numberOfSeats;  
    }  
  
    public String getLicensePlate() {  
        return this.licensePlate;  
    }  
}
```

Memanggil Method dari Superclass

```
public class Vehicle {  
    protected String licensePlate = null;  
  
    public void setLicensePlate(String license) {  
        this.licensePlate = license;  
    }  
}
```

```
public class Car extends Vehicle {  
  
    public void setLicensePlate(String license) {  
        super.setLicensePlate(license);  
    }  
  
}
```


Typecasting

- Suatu instance dari subclass dapat dijadikan sebagai instance dari superclassnya

```
Car car = new Car();  
Vehicle vehicle = car;
```

- Misal suatu instance dari class Car dapat dijadikan instance dari class Vehicle, karena Car diwariskan dari Vehicle atau bisa disebutkan Car adalah Vehicle

Upcasting dan Downcasting

- Upcasting
 - Melakukan casting sebuah object dari subclass ke superclassnya
- Downcasting
 - Melakukan casting sebuah object dari superclass ke subclassnya
 - Hanya valid jika object yang dituju adalah instance dari subclass yang sama
 - Lihat contoh

Downcasting

```
Car car = new Car();
```

```
// upcast to Vehicle  
Vehicle vehicle = car;
```

```
// downcast to car again,  
valid  
Car car2 = (Car) vehicle;
```

```
Truck truck = new Truck();
```

```
// upcast to Vehicle  
Vehicle vehicle = truck;
```

```
// downcast to car again,  
tidak valid  
Car car = (Car) vehicle;
```

Overriding Method

- Di dalam subclass bisa dilakukan override (mendefinisikan ulang) method yang sudah ditulis di superclass

```
package oop.inheritance2;

public class Vehicle {
    String licensePlate = null;

    public void setLicensePlate(String licensePlate) {
        this.licensePlate = licensePlate;
    }
}
```

```
package oop.inheritance2;

public class Car extends Vehicle {
    public void setLicensePlate(String license) {
        this.licensePlate = license.toLowerCase();
    }
}
```

@override Annotation

- Anotasi `@override`
 - menghasilkan warning dari compiler jika method yang mau diOverride ternyata tidak meng-override apapun karena code method aslinya di superclass sudah dihapus

```
package oop.inheritance2;  
  
public class Car extends Vehicle {  
    @Override  
    public void setLicensePlate(String license) {  
        this.licensePlate = license.toLowerCase();  
    }  
}
```

instanceof Instruction

- Instruksi `instanceof` dapat digunakan untuk mengecek apakah suatu objek merupakan instance dari suatu class

```
Car car = new Car();  
boolean isCar = car instanceof Car;
```

Fields and Inheritance

- Field / atribut suatu superclass akan diwariskan ke subclass
- Access modifier membatasi akses atribut
 - public dan protected : atribut superclass bisa diakses oleh subclass
 - default : atribut superclass bisa diakses oleh subclass yang berada pada package yang sama
 - private : atribut superclass tidak bisa diakses oleh subclass
 - <https://www.geeksforgeeks.org/access-modifiers-java/>
- Atribut tidak bisa di-override
 - Atribut pada subclass dengan nama yang sama dengan atribut pada superclass akan menutupi atribut pada superclass
 - Jika subclass mengakses atribut ini, yang diakses adalah atribut pada subclass
 - Jika subclass memanggil method yang ada di superclass, dan method tersebut mengakses atribut yang sama dengan atribut bernama sama pada subclass, atribut yang diakses adalah atribut pada subclass

Fields and Inheritance

```
package oop.inheritance2;

public class ParentClass {
    private int privateNumber = 0;
    int defaultNumber = 1;
    protected int protectedNumber = 2;
    public int publicNumber = 3;
}
```

```
package oop.inheritance2;

public class ChildClass extends ParentClass {
    int number1, number2, number3, number4;

    public ChildClass() {
        this.number2 = super.defaultNumber;
        this.number3 = super.protectedNumber;
        this.number4 = super.publicNumber;
        this.number1 = super.privateNumber;
    }
}
```

⚠ The field ParentClass.privateNumber is not visible
5 quick fixes available:

- ➡ [Change visibility of 'privateNumber' to 'protected'](#)
- ➡ [Create getter and setter for 'privateNumber'...](#)
- ➡ [Change to 'defaultNumber'](#)
- ➡ [Change to 'protectedNumber'](#)
- ➡ [Change to 'publicNumber'](#)

Constructors and Inheritance

- Constructor dari sebuah superclass tidak diwariskan ke subclass
- Subclass masih dapat mengakses constructor superclass menggunakan keyword super

Constructors and Inheritance

```
package oop.inheritance;

public class Vehicle {
    protected String licensePlate = null;

    // constructor
    public Vehicle() {
        this.licensePlate = "AD 1234 XY";
    }

    public void setLicensePlate(String license) {
        this.licensePlate = license;
    }
}
```

```
package oop.inheritance;

public class Car extends Vehicle {
    int numberOfSeats = 0;

    // constructor
    public Car() {
        // mengakses constructor superclass
        super();
    }

    public int getNumberOfSeats() {
        return this.numberOfSeats;
    }

    public String getLicensePlate() {
        return this.licensePlate;
    }

    public static void main(String[] args) {
        Car car = new Car();
        System.out.println("nomor plat : "+car.getLicensePlate());
    }
}
```

Nested Classes and Inheritance

- Aturan pewarisan nested class sesuai access modifier
- Private Nested classes tidak bisa diwariskan
- Nested classes dengan default (package) access modifier hanya bisa diakses oleh subclass yang berada di dalam package yang sama
- Nested classes dengan access modifier protected atau public selalu bisa diakses oleh subclass

Nested Classes and Inheritance



```
package oop.inheritance;

public class MyClass {
    //private access modifier
    private class MyNestedClass {
    }
}
```

```
package oop.inheritance;

public class MySubclass extends MyClass {
    public static void main(String[] args) {
        MySubclass subclass = new MySubclass();

        MyNestedClass nested = subclass.new MyNestedClass();
    }
}
```

 The type MyNestedClass is not visible
1 quick fix available:
 [Change visibility of 'MyNestedClass' to 'package'](#)

Nested Classes and Inheritance

```
package oop.inheritance;

public class MyClass {
    //default access modifier
    class MyNestedClass {

    }
}
```

```
package oop.inheritance;

public class MySubclass extends MyClass {
    public static void main(String[] args) {
        MySubclass subclass = new MySubclass();

        MyNestedClass nested = subclass.new MyNestedClass();
    }
}
```

Nested Classes and Inheritance

```
package oop.inheritance;

public class MyClass {
    //private access modifier
    private class MyNestedClass {
    }
}
```

```
package oop.inheritance2;

import oop.inheritance.MyClass;

public class MySubclass extends MyClass {
    public static void main(String[] args) {
        MySubclass subclass = new MySubclass();

        MyNestedClass nested = subclass.new MyNestedClass();
    }
}
```

 The type MyNestedClass is not visible

1 quick fix available:


 [Change visibility of 'MyNestedClass' to 'public'](#)

Final Classes and Inheritance

- Final class tidak bisa diwariskan

```
public final class MyFinalClass {  
    public static void main(String[] args) {  
        System.out.println("test final class");  
    }  
}
```

```
public class SubFinalClass extends MyFinalClass {  
}
```

 The type SubFinalClass cannot subclass the final class MyFinalClass
1 quick fix available:

 Remove 'final' modifier of 'MyFinalClass'

Abstract Classes and Inheritance

- Aturan pewarisan abstract class sama seperti class biasa

```
public abstract class MyAbstractProcess {  
    public void process() {  
        stepBefore();  
        action();  
        stepAfter();  
    }  
  
    public void stepBefore() {  
        //implementation directly in abstract superclass  
    }  
  
    public abstract void action(); // implemented by subclasses  
  
    public void stepAfter() {  
        //implementation directly in abstract superclass  
    }  
}
```

```
public class SubclassAbstrak extends MyAbstractProcess {  
  
    @Override  
    public void action() {  
        // TODO Auto-generated method stub  
    }  
}
```

- <http://tutorials.jenkov.com/java/abstract-classes.html>

- <http://tutorials.jenkov.com/java/inheritance.html>