PPT Praktikum PBO Week 9

Inheritance (Pewarisan)

Inheritance

- 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.

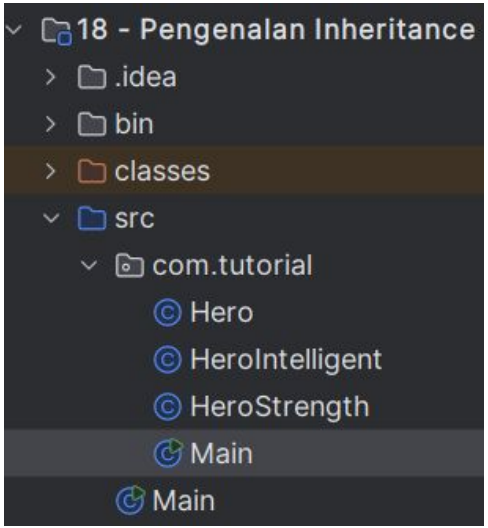
Kenapa Menggunakan Inheritance?

- Kode Dapat Digunakan Kembali: Kode yang ditulis di Superclass bersifat umum untuk semua subkelas. Kelas anak dapat langsung menggunakan kode kelas induk.
- Penggantian Metode: Penggantian Metode hanya dapat dicapai melalui Warisan. Ini adalah salah satu cara Java mencapai Run Time Polymorphism.
- Abstraksi: Konsep abstrak dimana kita tidak harus memberikan semua detail dicapai melalui pewarisan. Abstraksi hanya menampilkan fungsionalitas kepada pengguna.

Terminologi Dalam Inheritance

- Class: Class adalah sekumpulan objek yang memiliki karakteristik/perilaku yang sama dan properti/atribut yang sama. Kelas bukanlah entitas dunia nyata. Itu hanyalah sebuah templat atau cetak biru atau prototipe dari mana objek dibuat.
- SuperClass/Parent Class: Kelas yang fitur-fiturnya diwarisi dikenal sebagai superkelas (atau kelas dasar atau kelas induk).
- SubClass/Child Class: Kelas yang mewarisi kelas lain dikenal sebagai subkelas (atau kelas turunan, kelas diperluas, atau kelas anak). Subkelas dapat menambahkan bidang dan metodenya sendiri selain bidang dan metode superkelas.
- Reusability: Inheritance mendukung konsep “reusability”, yaitu ketika kita ingin membuat kelas baru dan sudah ada kelas yang memuat beberapa kode yang kita inginkan, kita bisa menurunkan kelas baru kita dari kelas yang sudah ada. Dengan melakukan ini, kami menggunakan kembali bidang dan metode kelas yang ada.

Hirarki Project



Class Hero

```java
package com.tutorial;

// superclass, parent class, base class
class Hero{  4 usages  2 inheritors
    String name;  4 usages

    // method
    void display() { System.out.println("Name : " + this.name); }
}
```

Class HeroIntelligent

```
1  package com.tutorial;
2
3  class HeroIntelligent extends Hero{  2 usages
4
5  }
```

Class HeroStrength

```
1  package com.tutorial;
2
3  // subclass, child class, derived class
4  class HeroStrength extends Hero{  2 usages
5
6  }
```

Class Main

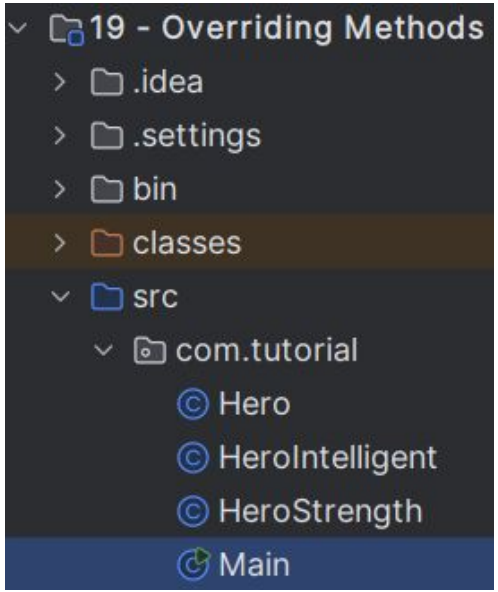
```java
package com.tutorial;

// inheritance adalah hubungan is-a
public class Main{
    public static void main(String[] args) {
        Hero hero1 = new Hero();
        hero1.name = "ucup";
        hero1.display();

        HeroStrength hero2 = new HeroStrength();
        hero2.name = "otong";
        hero2.display();

        HeroIntelligent hero3 = new HeroIntelligent();
        hero3.name = "Michael";
        hero3.display();
    }
}
```

Method Overriding

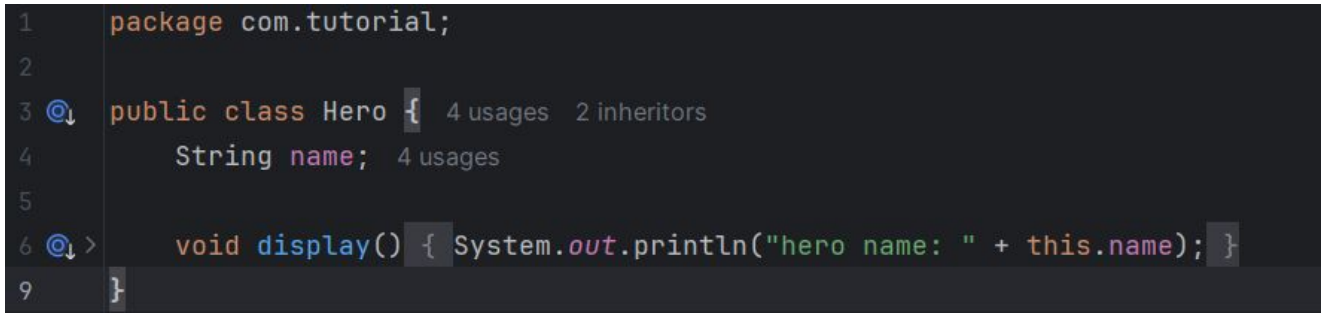
Method Overriding

- 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

Hirarki Project



Class Hero


package com.tutorial;
public class Hero { 4 usages 2 inheritors
String name; 4 usages
void display() { System.out.println("hero name: " + this.name); }
}

Class HeroIntelligent

```java
package com.tutorial;

public class HeroIntelligent extends Hero{  no usages

}
```

Class HeroStrength

```java
package com.tutorial;

public class HeroStrength extends Hero{  2 usages
    double defencePower;  2 usages

    // overriding methods
    void display(){  2 usages
        System.out.println("\nHero Strength");
        System.out.println("Hero Name : " + this.name);
        System.out.println("Defence Power : " + this.defencePower);
    }
}
```

Class HeroStrength

```java
package com.tutorial;

public class HeroStrength extends Hero{  2 usages
    double defencePower;  2 usages

    // overriding methods
    void display(){  2 usages
        System.out.println("\nHero Strength");
        System.out.println("Hero Name : " + this.name);
        System.out.println("Defence Power : " + this.defencePower);
    }
}
```

Class Main

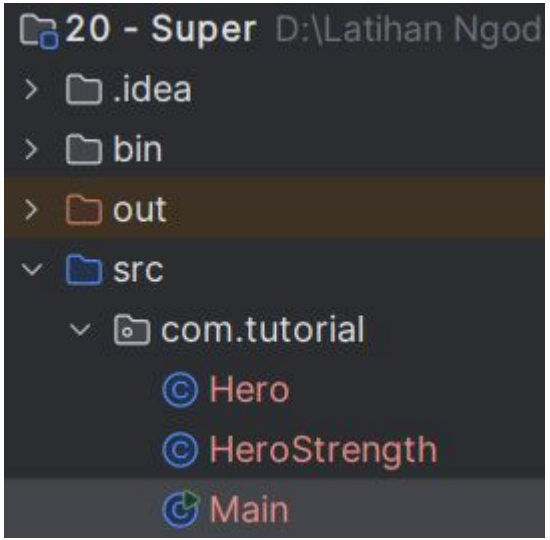
```java
package com.tutorial;

public class Main{
    public static void main(String[] args) {
        Hero hero1 = new Hero();
        hero1.name = "Saitama";
        // hero1.defencePower = gak punya;
        hero1.display();

        HeroStrength hero2 = new HeroStrength();
        hero2.name = "Mountain Lady";
        hero2.defencePower = 100;
        hero2.display();
    }
}
```

Super Keyword

Super Keyword

- 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

Hirarki Project



Class Hero

```java
package com.tutorial;

public class Hero {  3 usages  1 inheritor
    String name = "Class Hero";  2 usages

    void display() { System.out.println("Ini adalah " + this.name); }

    void dummyMethod() { System.out.println("method ini ada di superclass"); }
}
```

Class HeroStrength

```java
package com.tutorial;

// this.name akan mengakses atribut name kalau ada di kelas ini.
// jika tidak ada maka akan mengkases atribut name dari superclass.

// super.name akan selalu mengakses atribut superclass
public class HeroStrength extends Hero{  2 usages
    String name = "Class Strength";  1 usage

    void display(){  2 usages
        System.out.println("Ini adalah " + this.name);
        this.dummyMethod();
    }

    void displaySuper(){  1 usage
        System.out.println("Ini adalah " + super.name);
        super.dummyMethod();
    }

    void dummyMethod() { System.out.println("method ini ada di subclass"); }
}
```

Class Main

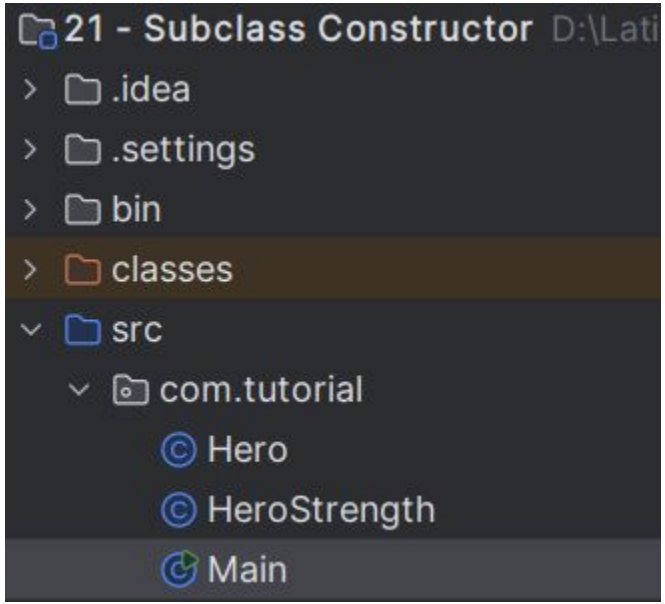
```java
package com.tutorial;

class Main{
    public static void main(String[] args) {
        Hero hero1 = new Hero();
        HeroStrength hero2 = new HeroStrength();

        hero1.display();
        hero2.display();
        hero2.displaySuper();
    }
}
```

Super Constructor

Super Constructor

- 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.

Hirarki Project



Class Hero

```java
package com.tutorial;

public class Hero {  1 usage  1 inheritor
    String name;  2 usages
    double defencePower;  2 usages
    double attackPower;  2 usages

    Hero(String name,double defencePower,double attackPower){  1 usage
        this.name = name;
        this.defencePower = defencePower;
        this.attackPower = attackPower;
    }

    void display(){  2 usages  1 override
        System.out.println("Name \t\t: " + this.name);
        System.out.println("Attack Power \t: " + this.attackPower);
        System.out.println("Defence Power \t: " + this.defencePower);
    }
}
```

Class HeroStrength

```java
package com.tutorial;

public class HeroStrength extends Hero{  2 usages
    String attribute = "Strength";  1 usage

    HeroStrength(String name, double defence, double attack){  1 usage
        super(name,defence,attack);
    }

    void display(){  2 usages
        System.out.println("Hero " + this.attribute);
        super.display();
    }
}
```

Class Main

```java
package com.tutorial;

class Main{
    public static void main(String[] args) {
        HeroStrength hero1 = new HeroStrength( name: "Naruko", defence: 50, attack: 10);
        hero1.display();
    }
}
```

Latihan

Program Game Sederhana

Pada pertemuan sebelumnya, kita sudah mencoba untuk membuat game sederhana tanpa menggunakan prinsip inheritance, pada latihan kali ini, kita akan mencoba untuk membuat game sederhana dengan menggunakan prinsip inheritance yang sudah kita pelajari. Program akan memiliki 3 class yang memiliki persyaratan sebagai berikut:

Class Hero:

Terdapat 3 atribut yaitu nama, attackPower, dan health. memiliki 3 fungsi, yaitu attack, takeDemage, dan display. Fungsi attack untuk menampilkan tampilan attack dan memproses serangan. Fungsi takeDemage digunakan untuk menampilkan demage yang diterima oleh lawan dan memproses proses pengurangan darah lawan. Fungsi display digunakan untuk menampilkan semua atribut dari Hero.

Program Game Sederhana

Class HeroStrength:

Merupakan anak dari class Hero. Memiliki nilai atribut tipe yang memiliki nilai default “Strenght”. Memiliki 2 fungsi overriding bernama display dan takeDemage. Fungsi display berisikan fungsi display pada kelas parent dan ditambahkan tampilan tipe hero. Fungsi takeDemage berisikan proses penyerangan dimana ketika heroStrength terkena demage, maka hanya setengah demage yang terkena olehnya.

Class Main:

Instansiasi 2 object yaitu hero1 dari class Hero dan hero2 dari class HeroStrength. Lakukan display awal kemudian lakukan 2 kali penyerangan, yaitu hero1 attack hero2 dan hero2 attack hero1 kemudian lakukan display untuk melihat perubahan.

Contoh Tampilan

```
Name: Tanjiro
Health: 100.0
Power: 10.0

Name: Zenitsu
Health: 100.0
Power: 20.0
Type : Strength

Tanjiro attack Zenitsu
Zenitsu receive half damage 10.0 -> 5.0

Zenitsu attack Tanjiro
Tanjiro receive damage 20.0

Name: Tanjiro
Health: 80.0
Power: 10.0

Name: Zenitsu
Health: 95.0
Power: 20.0
Type : Strength
```