# Tugas Hari ke-10

## Penjelasan materi tentang *Inherit, Override, Polymorph* & *Abstract*

1. Inheritance

Inheritance atau jika diartikan dalam bahasa Indonesia adalah Pewarisan. Inheritance adalah suatu konsep pemrograman dimana sebuah class (Parent Class) dapat mewariskan property dan method yang dimilikinya kepada class (Child Class) lain.

Namun tidak semua property dan method dari class parent akan diturunkan. Property dan method dengan hak akses private, tidak akan diturunkan kepada class child, hanya property dan method dengan hak akses protected dan public saja yang bisa diakses dari class child.

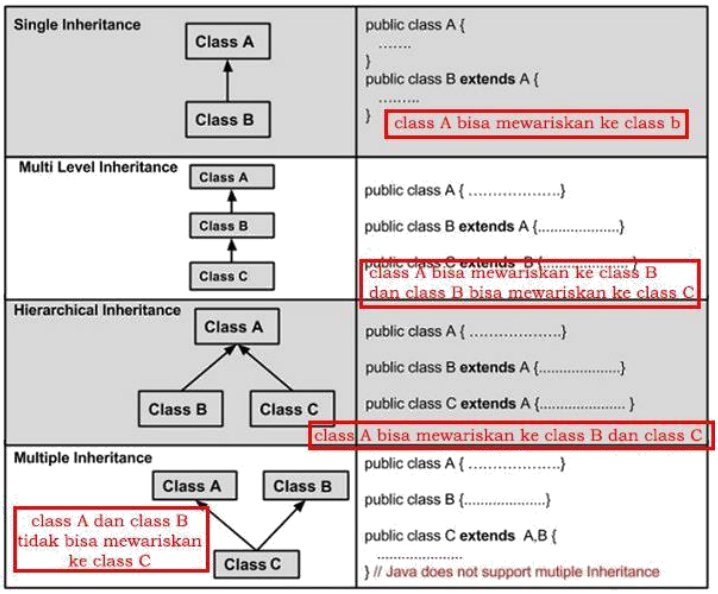
Pada class child, jika ingin mengakses class parent maka tambah property extends saat mendefinisikan class. Dalam inheritance, untuk merepresentasikan objek dari class parent, maka pada class child harus menggunakan super.

Untuk lebih jelas dalam memahami, bisa dilihat pada screenshot code di bawah ini yang sudah diberi komentar:

```java
//membuat class parent
class Calculation {
    //membuat variable z
    int z;

    //membuat method tambah dengan parameter x dan y
    //dengan operasi matematika pertambahan
    public void tambah(int x, int y) {
        z = x + y;
        System.out.println("Jumlah setelah ditambahkan : "+z);
    }

    //membuat method pengurangan
    //dengan operasi matematika pengurangan
    public void kurang(int x, int y) {
        z = x - y;
        System.out.println("Jumlah setelah dikurangkan : "+z);
    }
}
```

```java
//membuat class child yg di extends ke class parent
public class My_Calculation extends Calculation {

    //membuat method perkalian dengan parameter x dan y
    //dengan operasi matematika perkalian
    public void perkalian(int x, int y) {
        z = x * y;
        System.out.println("Jumlah perkalian :"+z);
    }

    //membuat method main, untuk running program
    public static void main(String args[]) {

        //menentukan variable a dan b yg akan dijadikan parameter
        int a = 20, b = 10;

        //buat objek
        My_Calculation demo = new My_Calculation();

        //jalankan
        demo.tambah(a, b);
        demo.kurang(a, b);
        demo.perkalian(a, b);
    }
}
```

Untuk lebih memahami tipikal pewarisan dapat diwariskan ke mana saja, bisa di lihat pada gambar di bawah ini:




2. Override

Override atauOverriding adalah sebuah fungsi atau method dari class parent yang ditulis kembali pada class child. Ada beberapa kaidah dalam penulisan, yaitu:

a. Parameter pada fungsi atau method overriding di class child harus sama dengan parameter yang terdapat pada class parent.
b. Tipe return pada class parent harus sama dengan class child.
c. Akses terhadap modifier class child tidak boleh melebihi tingkat akses modifier class parent.

Berikut contoh penerapan overriding pada program:

```
1  //membuat class Animal sebagai class parent
2  class Animal {
3
4      //membuat method move dan menampilkan hasil
5      public void move() {
6          System.out.println("Animals can move");
7      }
8  }
9
10 //membuat class Dog sebagai class child dengan menginduk kepada class Animal
11 class Dog extends Animal {
12
13     //membuat method move yg sama dengan method move yg ada pada class parent
14     public void move() {
15         System.out.println("Dogs can walk and run");
16     }
17 }
18
19 //membuat class untuk menjalankan program
20 public class TestDog {
21
22     //membuat method utama
23     public static void main(String args[]) {
24
25         //membuat objek
26         Animal a = new Animal();   // Animal reference and object
27         Animal b = new Dog();   // Animal reference but Dog object
28
29         //jalankan
30         a.move();   // runs the method in Animal class
31         b.move();   // runs the method in Dog class
32     }
33 }
34
```

3. Polymorph

Polymorph adalah sebuah prinsip di mana class dapat memiliki banyak “bentuk” method yang berbeda-beda meskipun namanya sama. “Bentuk” di sini dapat diartikan: isinya berbeda, parameternya berbeda, dan tipe datanya berbeda.

```java
//Class Test untuk mendemonstrasikan polymorphism
public class Test {

 //method main
 public static void main(String[] args) {

  //Mendeklarasikan array dengan tipe Manusia.
  Manusia [] manusia = new Manusia [4];

  //buat obejk
  manusia[0] = new Siswa();

  //jalankan
   manusia[0].makan();
   manusia[0].tidur();
   manusia[0].bergerak();
   System.out.println();
 }
}
```

```java
//membuat class Manusia sebagai class parent
public class Manusia {

 //method makan
 void makan(){
  System.out.println("Manusia makan");
 }

 //method tidur
 void tidur(){
  System.out.println("Manusia tidur");
 }

 //method bergerak
 void bergerak(){
  System.out.println("Manusia bergerak");
 }

}
```

```java
//membuat class Siswa sebagai class child
//yg menginduk pada class Manusia
public class Siswa extends Manusia {

 //override, method makan yg sama dengan method pada class Manusia
 @Override
 void makan(){
  System.out.println("Siswa makan");
 }

 //override, method tidur yg sama dengan method pada class Manusia
 @Override
 void tidur(){
  System.out.println("Siswa tidur");
 }

 //override, method bergerak yg sama dengan method pada class Manusia
 @Override
 void bergerak(){
  System.out.println("Siswa bergerak");
 }

}
```

4. Abstract

Abstract atau Abstraction merubahan sebuah class yang bisa dibilang setengah jadi yang memiliki method dan atribut. Class ini digunakan untuk membuat sebuah kelas yang memiliki method yang belum jelas implementasinya. Terdapat aturan dalam menerapkan pembuatan class abstrak, yaitu:

a. Jika sebuah class memiliki method abstrak maka class itu harus menjadi class abstrak.
b. Sebuah class abstrak dapat saja memiliki method yang tidak abstrak.
c. Jika sebuah class abstrak diturunkan menjadi class yang bukan abstrak, maka semua method abstrak dari class abstrak haruslah ditulis ulang / dibuat ulang di class yang bukan abstrak dan diberi detail dari methodnya.
d. Jika method abstrak di turunkan dan class turunannya adalah kelas abstrak, maka tidak perlu menulis ulang method yang abstrak.

```
1  //membuat class abstract
2  abstract class Employee {
3
4      //membuat atribut name, address, number
5      private String name;
6      private String address;
7      private int number;
8
9      //membuat contructor Employee dengan parameter name, address, number
10     public Employee(String name, String address, int number) {
11         System.out.println("Constructing an Employee");
12         this.name = name;
13         this.address = address;
14         this.number = number;
15     }
16
17     //membuat method computePay
18     public double computePay() {
19       System.out.println("Inside Employee computePay");
20       return 0.0;
21     }
22
23     //membuat method mailCheck
24     public void mailCheck() {
25         System.out.println("Mailing a check to " + this.name + " " + this.address);
26     }
27
28     //membuat method toString
29     public String toString() {
30         return name + " " + address + " " + number;
31     }
32
33     //membuat method getName
34     public String getName() {
35         return name;
36     }
37
```

```
37
38      //membuat method getAddress
39      public String getAddress() {
40          return address;
41      }
42
43      //membuat method setAddress
44      public void setAddress(String newAddress) {
45          address = newAddress;
46      }
47
48      //membuat method getNumber
49      public int getNumber() {
50          return number;
51      }
52  }
53
54  //membuat class AbstractDemo untuk menampung method main
55  public class AbstractDemo {
56
57      //membuat method main
58      public static void main(String [] args) {
59
60          //membuat objek
61          Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00);
62          Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00);
63          System.out.println("Call mailCheck using Salary reference --");
64          s.mailCheck();
65          System.out.println("\n Call mailCheck using Employee reference--");
66          e.mailCheck();
67      }
68  }
```

```
70  //membuat class Salary yang menginduk pada class Employee (class abstract)
71  //class Salary berfungsi untuk menjelaskan class Employee yg bersifat abstract
72  class Salary extends Employee {
73
74      //membuat atriibut salary
75      private double salary;
76
77      //membuat method salary
78      public Salary(String name, String address, int number, double salary) {
79          super(name, address, number);
80          setSalary(salary);
81      }
82
83      //membuat method mailCheck
84      public void mailCheck() {
85          System.out.println("Within mailCheck of Salary class ");
86          System.out.println("Mailing check to " + getName() + " with salary " + salary);
87      }
88
89      //membuat method getSalary
90      public double getSalary() {
91          return salary;
92      }
93
94      //membuat method setSalary
95      public void setSalary(double newSalary) {
96          if(newSalary >= 0.0) {
97              salary = newSalary;
98          }
99      }
100
101     //membuat method computePay
102     public double computePay() {
103         System.out.println("Computing salary pay for " + getName());
104         return salary/52;
105     }
106 }
```

5. Encapsulation

Encapsulation adalah membungkus class dan menjaga apa saja yang ada di dalam class tersebut, baik method ataupun atribut, agar tidak dapat di akses oleh class lainnya. Untuk menjaga hal tersebut bisa menggunakan hak akses modifier, yang terdiri dari :

a. Private
   Memberikan hak akses hanya pada class itu sendiri, artinya apa saja yang ada di dalam class A baik itu method ataupun atribut hanya bisa diakses oleh class A saja, class lain tidak bisa mengaksesnya.
b. Public
   Memberikan hak akses kepada atribut atau method agar bisa diakses oleh siapapun (property atau class lain diluar class yang bersangkutan), artinya method atau atribut yang ada di class A dapat di akses oleh siapaun baik itu class A, class B dan seterusnya.
c. Protected
   Memberikan hak akses kepada class itu sendiri dan class hasil turunannya (inheritance), artinya apa saja yang ada di class A hanya bisa di akses oleh class A sendiri dan class yang mewarisi class A. Namun harus dipahami class lain yang berada dalam satu package dengan class A mampu mengakses tipe data protected, sedangkan yang tidak mampu mengakses adalah class-class yang berada diluar package class A. Untuk dapat mengaksesnya, class yang berada di luar package class A harus mewarisi class A.

Agar lebih jelas dalam penggunaan, bisa di lihat pada penerapan dalam program berikut:

```
1   //membuat class Belajar
2   class Belajar{
3
4     //membuat atribut dengan modifier
5     public String x ="Pintar";        //modifier public
6     protected String y = "Java";      //modifier protected
7   }
8
9   //membuat class Pintar yg berisi method main
10  //dan menggunakan atribut class Belajar
11  public class Pintar{
12
13    //membuat method main
14    public static void main(String[]args){
15
16    //buat objek
17    Belajar panggil = new Belajar();
18
19    //cetak
20    System.out.println("X adalah modifier public");
21    System.out.println("Y adalah modifier protected");
22
23    System.out.println("Panggil X dari kelas lain, X = "+panggil.x);
24    System.out.println("Y tidak bisa dipanggil karena modifier bersifat protected");
25    }
26  }
27
```