

## TUGAS SESI 3

### PRATIUM PERCOBAAN INHERITANCE



Nama : Ranti indriyani

Kelas : TI22C

NIM : 20220040002

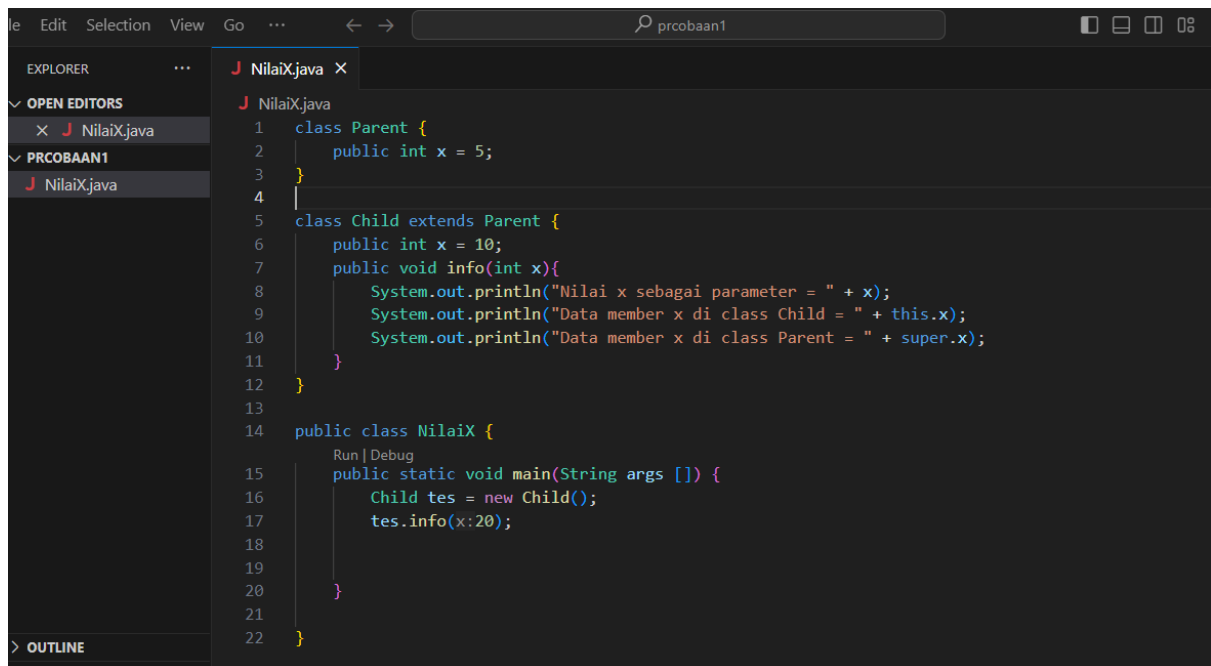
Dosen Pengampu : Alun Sujjada, S.Kom, M.T

Jurusan Teknik Informatika

UNIVERSITAS NUSA PUTRA SUKABUM

## PERCOBAAN 1

Percobaan berikut ini menunjukkan penggunaan kata kunci “super”.



```
1 class Parent {
2     public int x = 5;
3 }
4
5 class Child extends Parent {
6     public int x = 10;
7     public void info(int x){
8         System.out.println("Nilai x sebagai parameter = " + x);
9         System.out.println("Data member x di class Child = " + this.x);
10        System.out.println("Data member x di class Parent = " + super.x);
11    }
12 }
13
14 public class NilaiX {
15     public static void main(String args []) {
16         Child tes = new Child();
17         tes.info(x:20);
18     }
19 }
20
21
22 }
```

Penjelasannya :

### 1. Kelas :

- Kode mendefinisikan tiga kelas: , , dan .ParentChildNilaiX
- Kelas adalah cetak biru untuk membuat objek yang berbagi properti serupa (anggota data) dan perilaku (metode).

### 2. Warisan:

- Kelas memperluas kelas. Ini berarti mewarisi semua anggota dan metode data (kecuali secara eksplisit diganti) dari .ChildParentChildParent
- Pewarisan memungkinkan penggunaan kembali kode dan spesialisasi.

### 3. Anggota Data (x):

- Keduanya dan kelas memiliki anggota data bernama .ParentChildx
- Dalam , diinisialisasi menjadi .Parentx5
- Dalam , juga dideklarasikan, tetapi nilainya diatur ke . Ini menciptakan efek bayangan, di mana kelas anak menyembunyikan yang diwarisi dari kelas induk di dalam kelas itu sendiri.Childx10xChild

### 4. Info Metode:

- Kelas mendefinisikan metode publik yang disebut yang mengambil parameter integer .Childinfox
- Di dalam, tiga pernyataan digunakan: infoSystem.out.println
- Yang pertama mencetak nilai parameter yang diteruskan ke metode (20 dalam hal ini).x

- Yang kedua (menggunakan ) mencetak nilai dalam kelas (10), menunjukkan bayangan.this.xxChild
- Yang ketiga (menggunakan ) secara eksplisit mengakses yang diwarisi dari kelas (5). mengacu pada kelas induk.super.xxParentsuper

#### 5. NilaiX Kelas (Program Utama):

- Kelas ini berfungsi sebagai titik masuk untuk eksekusi program.
- Di dalam metode: main
- Sebuah objek bernama dibuat, memanggil konstruktor kelas.ChildtesChild
- Metode objek dipanggil, meneruskan nilai sebagai argumen.infotes20

Hasil :

Ketika Anda menjalankan kode ini, output berikut akan ditampilkan:

Nilai x sebagai parameter = 20

Data member x di class Child = 10

Data member x di class Parent = 5

## PERCOBAAN 2

Pertanyaanya seperti ini dan sebelum di rubah sintax nya kodingannya error seperti ini

```

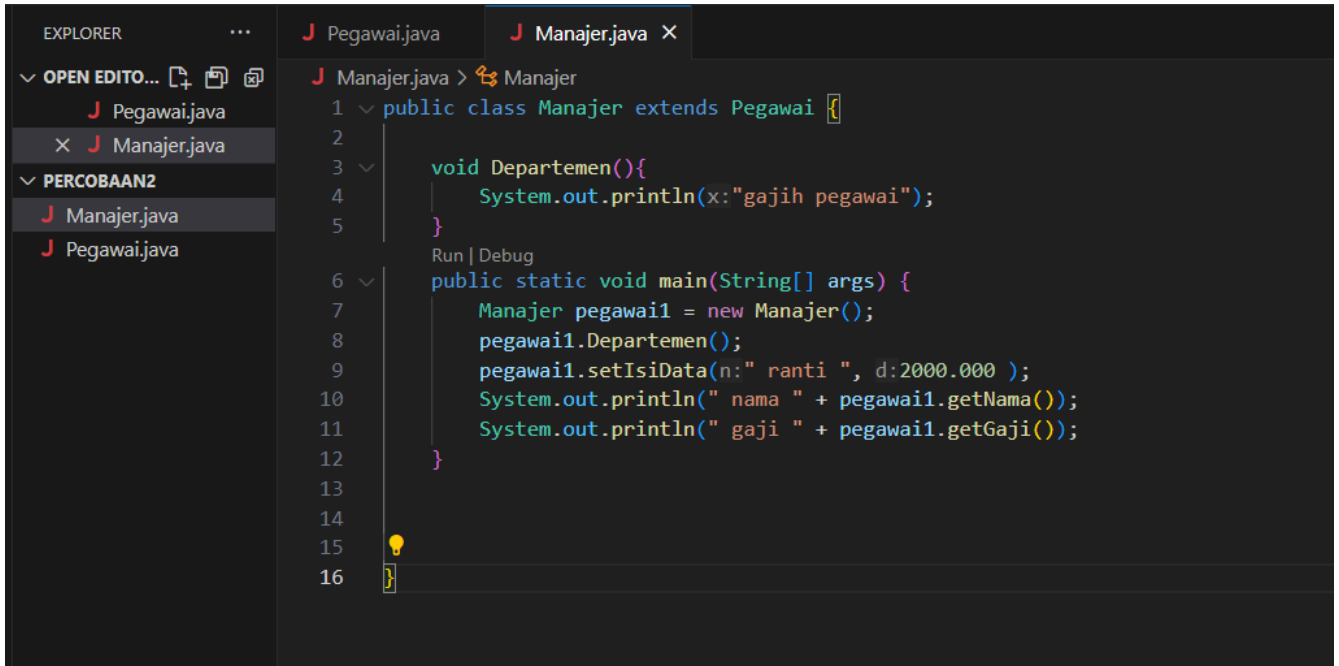
1 public class Pegawai {
2     private String nama;
3     public double gaji;
4 }
5
6
7 public class Manajer extends Pegawai {
8     public String departemen;
9
10    public void IsiData(String n, String d) {
11        nama=n;
12        departemen=d;
13    }
14 }
15

```

Setelah di syntax nya di rubah dan hasilnya seperti ini

Kita buat 2 file

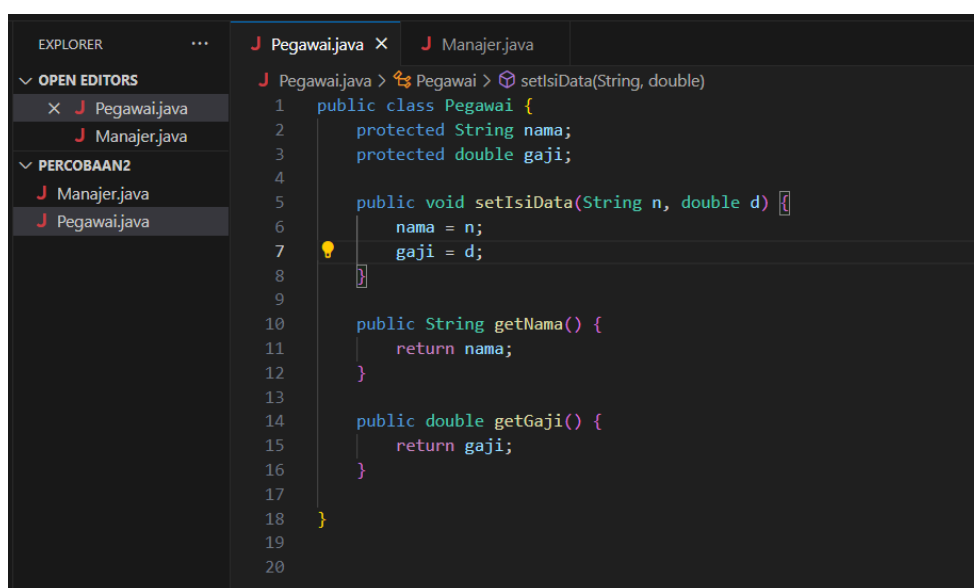
Ini file yang “Manajer”



```
1 public class Manajer extends Pegawai {
2
3     void Departemen(){
4         System.out.println(x:"gajih pegawai");
5     }
6
7     public static void main(String[] args) {
8         Manajer pegawai1 = new Manajer();
9         pegawai1.Departemen();
10        pegawai1.setIsiData(n:" ranti ", d:2000.000 );
11        System.out.println(" nama " + pegawai1.getNama());
12        System.out.println(" gaji " + pegawai1.getGaji());
13    }
14
15
16 }
```

Dan ini file

“PEGAWAI”



```
1 public class Pegawai {
2     protected String nama;
3     protected double gaji;
4
5     public void setIsiData(String n, double d) {
6         nama = n;
7         gaji = d;
8     }
9
10    public String getNama() {
11        return nama;
12    }
13
14    public double getGaji() {
15        return gaji;
16    }
17
18 }
19
20 }
```

Penjelasannya :

## Class Pegawai:

### a. Atribut

- nama: String - Menyimpan nama pegawai.
- gaji: Double - Menyimpan gaji pegawai.

### b. Metode

- setLsiData(String n, double d): Mengatur nilai nama dan gaji dengan nilai yang diberikan.
- getNama(): Mengembalikan nilai nama.
- getGaji(): Mengembalikan nilai gaji.

### c. Class Manajer

#### 1. Kelas turunan dari kelas Pegawai.

#### 2. Metode

- Departemen(): Mencetak "gajih pegawai".
- main(String[] args):
  - Membuat objek baru dari kelas Manajer dengan nama pegawai1.
  - Memanggil metode Departemen() pada objek pegawai1.
  - Mengatur nilai nama dan gaji objek pegawai1 dengan metode setLsiData("ranti ", 2000.000).
  - Mencetak nilai nama dan gaji objek pegawai1 dengan metode getNama() dan getGaji().

## Penjelasan kode:

1. Kode ini mendefinisikan dua kelas: Pegawai dan Manajer.
2. Kelas Pegawai memiliki dua atribut: nama dan gaji, dan tiga metode: setLsiData(), getNama(), dan getGaji().
3. Kelas Manajer mewarisi semua atribut dan metode dari kelas Pegawai.
4. Kelas Manajer memiliki satu metode tambahan: Departemen().
5. Metode main() di kelas Manajer membuat objek baru dari kelas Manajer, memanggil metode Departemen(), mengatur nilai nama dan gaji, dan mencetak nilai nama dan gaji.

## Kesimpulannya :

Kode ini mendemonstrasikan cara membuat kelas, mewarisi kelas, dan mendefinisikan metode dalam bahasa pemrograman Java.

### PERCOBAAN 3

Percobaan berikut ini menunjukkan penggunaan konstruktor yang tidak diwariskan.  
Mengapa terjadi error, dan bagaimana solusinya ?

```
J Parent.java > Child > Child()
1  public class Parent {
2      // kosong
3
4  }
5
6  public class Child extends Parent {
7      int x;
8      public Child() {
9          x = 5;
10     }
11 }
12
```

Keterangan :

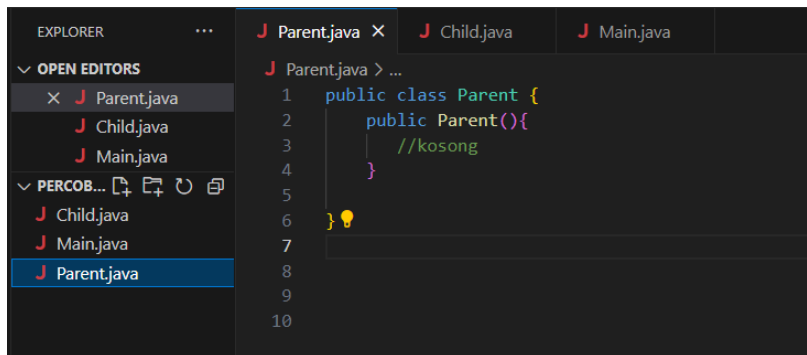
Kode menunjukkan error karena konstruktor tidak diwariskan dalam Java. Ketika Anda mendeklarasikan kelas Child yang mewarisi dari Parent, konstruktor dari kelas Parent tidak secara otomatis diwariskan ke kelas Child.

Alasan error:

1. Ketiadaan konstruktor di kelas Parent: Kelas Parent tidak memiliki konstruktor, baik default (tanpa parameter) maupun eksplisit (dengan parameter).
2. Ketidakcocokan konstruktor di kelas Child: Kelas Child memiliki konstruktor tanpa parameter, namun kelas Parent tidak memiliki konstruktor yang cocok untuk dipanggil.

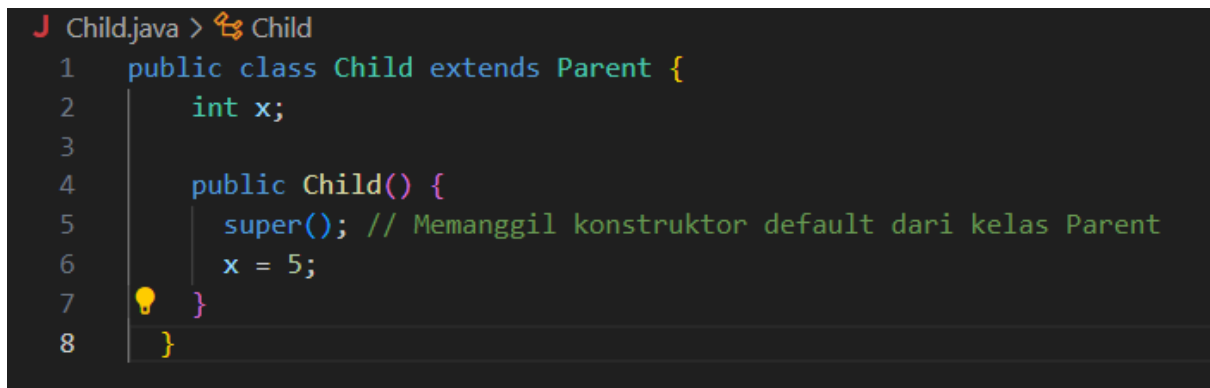
Solusi:

- a. Menambahkan konstruktor default di kelas Parent:

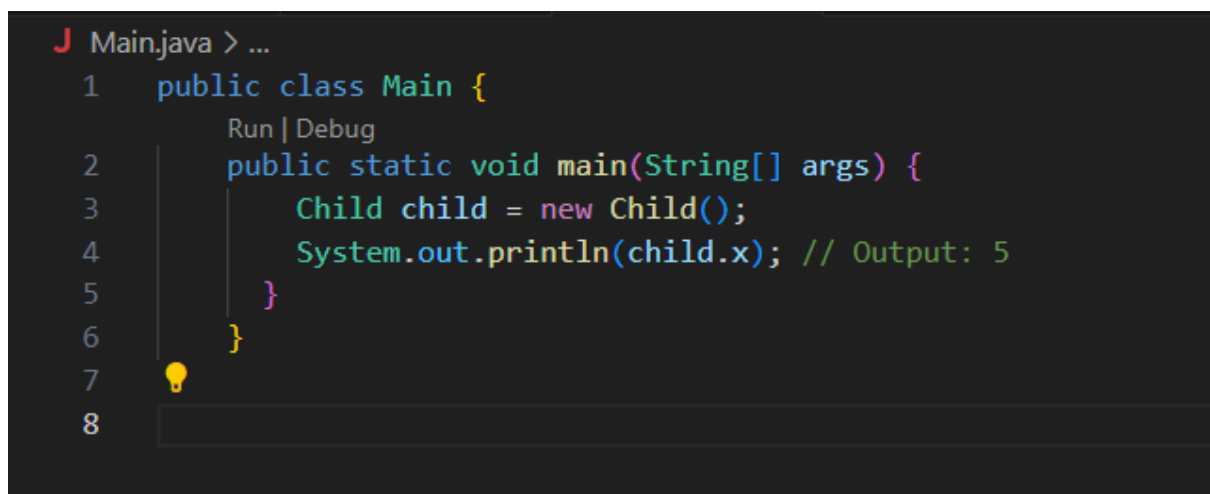


```
1 public class Parent {
2     public Parent(){
3         //kosong
4     }
5 }
6
7
8
9
10
```

- b. Memanggil konstruktor Parent di kelas Child:



```
1 public class Child extends Parent {
2     int x;
3
4     public Child() {
5         super(); // Memanggil konstruktor default dari kelas Parent
6         x = 5;
7     }
8 }
```



```
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         Child child = new Child();
5         System.out.println(child.x); // Output: 5
6     }
7 }
8
```

Penjelasan :

1. Class Parent:

- Berfungsi sebagai kelas dasar atau kerangka umum untuk kelas Child.
- Memiliki konstruktor default, yaitu konstruktor tanpa parameter yang tidak melakukan apa-apa dalam kode ini.

2. Class Child:

- Mewarisi sifat-sifat dari kelas Parent.
- Memiliki atribut x bertipe integer.
- Konstruktor Child melakukan dua hal :
  - a. Memanggil konstruktor default dari kelas Parent menggunakan `super()` untuk memastikan inisialisasi yang proper.
  - b. Menginisialisasi nilai x menjadi 5.

3. Class Main:

- Berisi metode `main()` yang merupakan pintu masuk program Java.
- Di dalam `main()`:
  - a. Dibuat objek baru dari kelas Child bernama `child`.
  - b. Nilai x dari objek `child` dicetak ke layar, menghasilkan output 5.

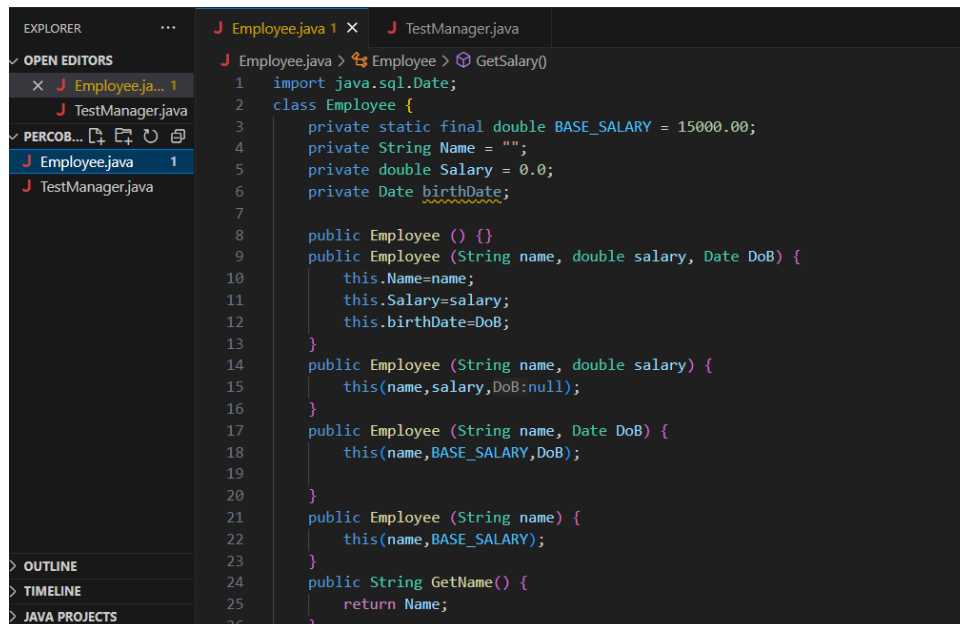
Kesimpulan :

1. Kode ini menunjukkan konsep inheritance atau pewarisan dalam Java, di mana kelas Child mewarisi dari kelas Parent.
2. Konstruktor Child memanggil konstruktor Parent untuk memastikan inisialisasi yang tepat.
3. Metode `main()` menciptakan objek Child dan berinteraksi dengannya untuk menghasilkan output.

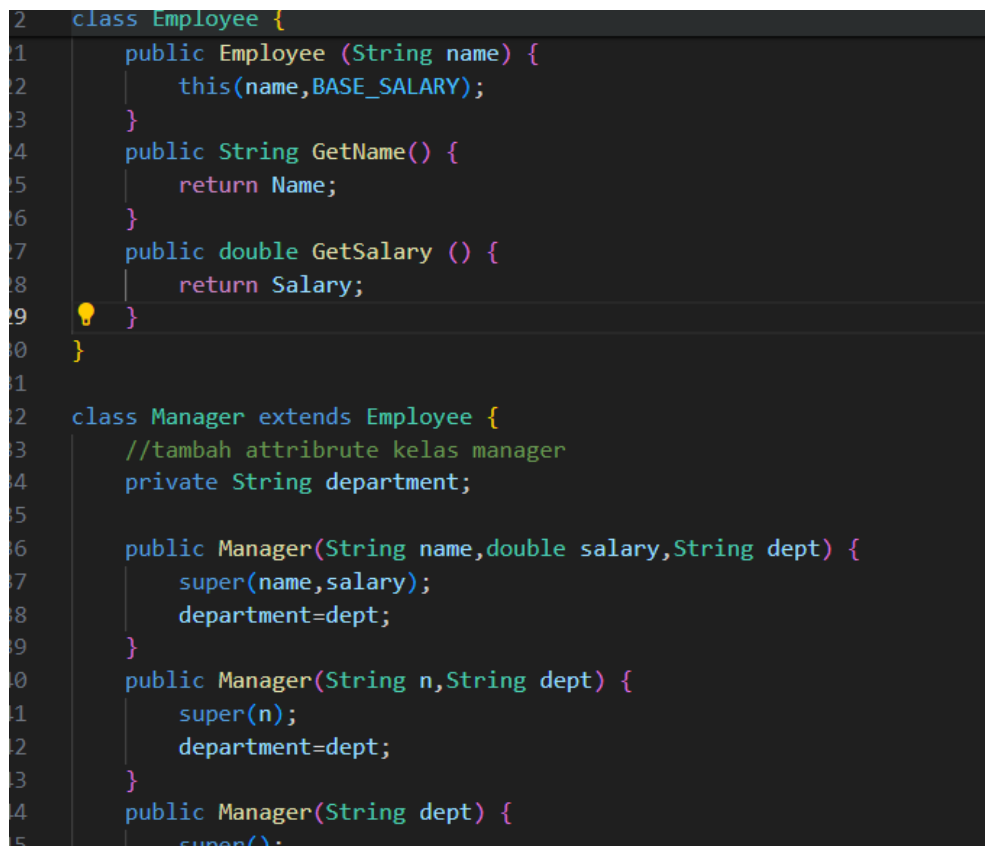


## PERCOBAAN 4

Percobaan berikut ini menunjukkan penggunaan kelas Employee dan subkelas Manager yang merupakan turunannya. Kelas TestManager digunakan untuk menguji kelas Manager



```
1 import java.sql.Date;
2 class Employee {
3     private static final double BASE_SALARY = 15000.00;
4     private String Name = "";
5     private double Salary = 0.0;
6     private Date birthDate;
7
8     public Employee () {}
9     public Employee (String name, double salary, Date DoB) {
10         this.Name=name;
11         this.Salary=salary;
12         this.birthDate=DoB;
13     }
14     public Employee (String name, double salary) {
15         this(name,salary,DoB:null);
16     }
17     public Employee (String name, Date DoB) {
18         this(name,BASE_SALARY,DoB);
19     }
20 }
21 public Employee (String name) {
22     this(name,BASE_SALARY);
23 }
24 public String GetName() {
25     return Name;
26 }
```



```
2 class Employee {
1     public Employee (String name) {
2         this(name,BASE_SALARY);
3     }
4     public String GetName() {
5         return Name;
6     }
7     public double GetSalary () {
8         return Salary;
9     }
10 }
11
12 class Manager extends Employee {
13     //tambah attribrute kelas manager
14     private String department;
15
16     public Manager(String name,double salary,String dept) {
17         super(name,salary);
18         department=dept;
19     }
20     public Manager(String n,String dept) {
21         super(n);
22         department=dept;
23     }
24     public Manager(String dept) {
25         super().
```

```

J TestManager.java > TestManager
1  public class TestManager {
2
3      Run | Debug
4      public static void main(String[] args) {
5          Manager Utama = new Manager(name:"John",salary:5000000,dept:"Financial");
6          System.out.println("Name:" + Utama.GetName());
7          System.out.println("Salary:" + Utama.GetSalary());
8          System.out.println("Department:" + Utama.GetDept());
9
10         Utama = new Manager(n:"Michael",dept:"Accounting");
11         System.out.println("Name:" +Utama.GetName());
12         System.out.println("Salary:" + Utama.GetSalary());
13         System.out.println("Department:" + Utama.GetDept());
14     }
15 }

```

Penjelasan :

1. Class Karyawan :
  - a. Mewakili karyawan dengan atribut seperti nama, gaji, dan tanggal lahir (jenis yang diimport dari).Datejava.sql
  - b. Menyediakan konstruktor untuk membuat objek: Employee
    - Konstruktor kosong (Employee())
    - Konstruktor dengan nama, gaji, dan tanggal lahir (Employee(String name, double salary, Date DoB))
    - Konstruktor dengan variasi pada parameter ini, menggunakan nilai default jika berlaku.
  - c. Metode untuk mengakses informasi karyawan
    - GetName(): Mengembalikan nama karyawan.
    - GetSalary(): Mengembalikan gaji karyawan.
2. Class Manajer:
  - a. Mewarisi dari kelas, artinya ia memiliki semua atribut dan metode .EmployeeEmployee
  - b. Menambahkan atribut baru (Jenis string) untuk mewakili departemen manajer.department
  - c. Menyediakan konstruktor untuk membuat objek: Manager
    - Konstruktor dengan nama, gaji, dan departemen (Manager(String name, double salary, String dept))
    - Konstruktor dengan variasi pada parameter ini, memanfaatkan pewarisan dari .Employee
  - d. Metode untuk mengakses departemen manajer
    - GetDept(): Mengembalikan departemen manajer.
3. Class TestManager:
  - a. Kelas ini menunjukkan cara membuat dan menggunakan objek.Manager
  - b. Dalam metode: main
    - Membuat objek bernama dengan detail seperti nama, gaji, dan departemen.ManagerUtama
    - Mencetak informasi tentang menggunakan metode pengambilnya.Utama
    - Membuat objek lain dengan detail berbeda dan menetapkan ke . Ini menimpa informasi untuk manajer pertama (tidak ideal).ManagerUtama
    - Mencetak informasi tentang objek yang ditimpa.Utama

## PERCOBAAN 5

Percobaan berikut ini menunjukkan penggunaan kelas MoodyObject dengan subkelas HappyObject dan SadObject. Kelas MoodyTest digunakan untuk menguji kelas dan subkelas.

- SadObject berisi :
  - a. sad, method untuk menampilkan pesan, tipe public
- HappyObject berisi :
  - b. laugh, method untuk menampilkan pesan, tipe public
- MoodyObject berisi :
  - c. getMood, memberi nilai mood sekarang, tipe public, return type string
  - d. speak, menampilkan mood, tipe public.

Kita buat 4 file  
class  
"MoodyObject"

Keterangan :

- Kelas ini mendefinisikan perilaku moody dasar (selalu mengatakan "moody").
- Kelas lain dapat mewarisi dari yang satu ini untuk membuat karakter murung tertentu yang berpotensi tertawa atau menangis nanti.



```
1 public class MoodyObject {
2     protected String getMood () {
3         return "moody";
4     }
5 }
6 public void speak() {
7     System.out.println("I am " +getMood());
8 }
9 void laugh() {}
10 void cry () {}
11 }
12
13
14
```

- Kelas ini mewarisi dari , artinya ia mendapatkan semua kemampuan (metode) dari .MoodyObjectMoodyObject
- Ini mendefinisikan ulang metode: getMood
- Ini mengimplementasikan metode: cry
-

```

J SadObject.java > SadObject
1 public class SadObject extends MoodyObject {
2     protected String getMood () {
3         return "sad";
4     }
5     public void cry() {
6         System.out.println(x:"Hoo hoo");
7     }
8 }

```

- Kelas ini mewarisi dari , yang berarti ia mendapatkan semua kemampuan moody dasar.MoodyObject
- Ini mendefinisikan ulang metode: getMood
- Ini mengimplementasikan metode: laugh

```

J HappyObject.java X J MoodyObject.java J SadObject.java
J HappyObject.java > HappyObject
1 public class HappyObject extends MoodyObject {
2     protected String getMood() {
3         return"happy";
4     }
5     public void laugh () {
6         System.out.println(x:"hahaha");
7     }
8 }

```

- Menguji kelas induk:  
Ini menciptakan MoodyObject (template moody dasar).  
Ia meminta MoodyObject untuk berbicara, mengharapkan respons "moody".
- Menguji HappyObject:  
Ini menciptakan HappyObject (karakter ceria).  
Ia meminta HappyObject untuk berbicara, mengharapkan respons "bahagia".  
Itu membuat HappyObject tertawa dengan suara "hahaha".
- Menguji SadObject:  
Ini menciptakan SadObject (karakter sedih).

Ia meminta SadObject untuk berbicara, mengharapkan respons "sedih". Ini memungkinkan SadObject menangis dengan suara "Hoo hoo".

Secara sederhana:

- Kelas ini menciptakan berbagai jenis karakter murung dan menguji bagaimana mereka berbicara, tertawa, atau menangis. Ini seperti bermain dengan suasana hati yang berbeda untuk melihat bagaimana mereka mengekspresikan diri!

```
J MoodyTest.java > MoodyTest > main(String[])
1  public class MoodyTest {
2
3      Run | Debug
4      public static void main(String[] args) {
5
6          MoodyObject m = new MoodyObject();
7
8          //test perent class
9          m.speak();
10
11
12          //test inheritance class
13          m = new HappyObject ();
14          m.speak();
15          m.laugh();
16
17          //test inheritance class
18          m=new SadObject();
19          m.speak();
20          m.cry();
21
22
23      }
24  }
```

## PERCOBAAN 6

Percobaan berikut ini menunjukkan penggunaan kelas A dan dengan subkelas B. Simpan kedua kelas ini dalam 2 file yang berbeda (A.java dan B.java) dan dalam satu package.

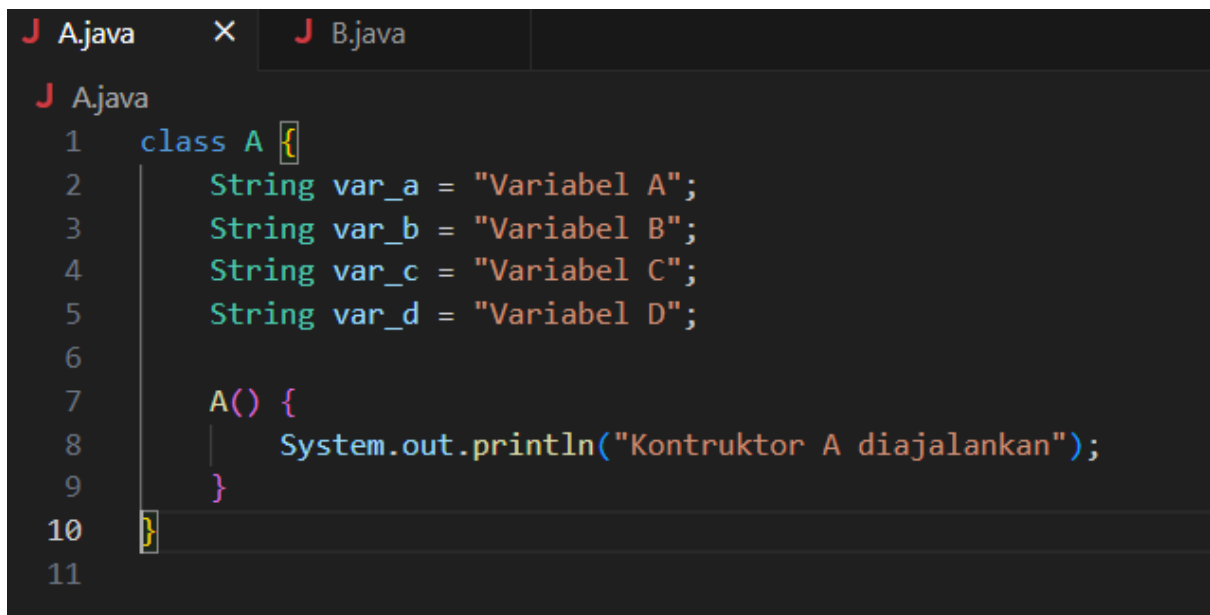
Perhatikan proses pemanggilan konstruktor dan pemanggilan variabel

Class A:

- a. Ini adalah sebuah class bernama "A".
- b. Class ini memiliki 4 variabel
  - var\_a dengan nilai "Variabel A".
  - var\_b dengan nilai "Variabel B".
  - var\_c dengan nilai "Variabel C".
  - var\_d dengan nilai "Variabel D".
- c. Class ini memiliki konstruktor bernama A().
- d. Konstruktor ini akan dijalankan ketika sebuah object baru dari class "A" dibuat.
- e. Konstruktor ini hanya mencetak pesan "Kontruktor A diajalankan".

Singkatnya :

Class ini mendefinisikan sebuah "template" untuk membuat object dengan 4 variabel dan sebuah konstruktor.



```
J A.java X J B.java
J A.java
1  class A {
2      String var_a = "Variabel A";
3      String var_b = "Variabel B";
4      String var_c = "Variabel C";
5      String var_d = "Variabel D";
6
7      A() {
8          System.out.println("Kontruktor A diajalankan");
9      }
10 }
11
```

Class B:

- a. Ini adalah class "B" yang "mewarisi" semua yang ada pada class "A".
- b. Class ini menambahkan sebuah konstruktor bernama B().
- c. Ketika sebuah object "B" dibuat, konstruktor ini:
  - o Mencetak pesan "Kontruktor B diajalankan".
  - o Mengubah nilai var\_a dan var\_b dengan nilai baru khusus untuk object "B".

Metode main:

- a. Ini adalah bagian yang menjalankan kode.
- b. Ia membuat dua object:
  - o aa dari class "A".
  - o bb dari class "B".
- c. Ia kemudian menampilkan nilai-nilai variabel dari kedua object tersebut.

Yang perlu diperhatikan:

Nilai var\_a dan var\_b berbeda untuk object "A" dan "B" karena class "B" mengubahnya pada konstruktornya.

Nilai var\_c dan var\_d tetap sama untuk keduanya karena tidak diubah pada class "B".

Singkatnya :

- o Class "B" mewarisi semua yang ada pada class "A", namun bisa mengubah beberapa hal untuk object-object "B" yang dibuat.
- o Kode ini menunjukkan bagaimana pewarisan bekerja untuk object dan variabel.

```
J B.java > B
1  class B extends A {
2      B () {
3          System.out.println(x:"Kontruktor B dijalankan ");
4          var_a = "var_a dari class B";
5          var_b = "Var_a dari class B";
6      }
7
8      Run | Debug
9      public static void main(String args[]) {
10
11          System.out.println(x:"Object A dibuat");
12          A aa= new A();
13          System.out.println(x:"menampilkan nama variabel object aa");
14          System.out.println(aa.var_a);
15          System.out.println(aa.var_b);
16          System.out.println(aa.var_c);
17          System.out.println(aa.var_d);
18          System.out.println(x:"");
19
20          System.out.println(x:"Object B dibuat ");
21          B bb= new B();
22          System.out.println(x:"menampilakan nama variabel objek bb");
23          System.out.println(bb.var_a);
24          System.out.println(bb.var_b);
25          System.out.println(bb.var_c);
26          System.out.println(bb.var_d);
27      }
```

## PERCOBAAN 7

Percobaan berikut ini menunjukkan penggunaan Inheritance dan Overriding method pada kelas Bapak dan subkelas Anak. Terjadi override pada method show\_variabel. Perhatikan perubahan nilai pada variabel a, b, dan c

SEBELUM DI RUBAH SINTAX NYA SEPERTI INI

```
Bapak.java > Bapak > show_variabel()
1  class Bapak {
2      int a;
3      int b;
4      void show_variabel() {
5          System.out.println("Nilai a = " + a);
6          System.out.println("Nilai b = " + b);
7      }
8  }
9
10 class Anak extends Bapak{
11     int c;
12     void show_variabel(){
13         System.out.println("Nilai a = " + a);
14         System.out.println("Nilai b = " + b);
15         System.out.println("Nilai c = " + c);
16     }
17 }
18
19 public class InheritExample {
20     Run | Debug
21     public static void main(String[] args) {
22         Bapak objectBapak = new Bapak();
23         Anak objectAnak = new Anak();
24
25         objectBapak.a=1;
26
27         objectBapak.a=1;
28         objectBapak.b=1;
29         System.out.println(x:"Object Bapak (Superclass) :");
30
31         objectBapak.show_variabel();
32         objectAnak.c=5;
33         System.out.println(x:"Object Anak (Superclass dari Bapak) :");
34         objectAnak.show_variabel();
35     }
36 }
```



jadi seperti ini

```
J Bapak.java > ...
1  class Bapak {
2      int a;
3      int b;
4      void show_variabel() {
5          System.out.println("Nilai a = " + a);
6          System.out.println("Nilai b = " + b);
7      }
8  }
9
10 class Anak extends Bapak{
11     int c;
12     void show_variabel(){
13         System.out.println("Nilai a = " + a);
14         System.out.println("Nilai b = " + b);
15         System.out.println("Nilai c = " + c);
16     }
17 }
18
19
```

```
J InheritExample.java > Run InheritExample > main(String[])
1  public class InheritExample {
2      public static void main(String[] args) {
3
4          Bapak objectBapak = new Bapak();
5          Anak objectAnak = new Anak();
6
7          objectBapak.a=1;
8          objectBapak.b=1;
9          System.out.println(x:"Object Bapak (Superclass) :");
10
11         objectBapak.show_variabel();
12         objectAnak.c=5;
13         System.out.println(x:"Object Anak (Superclass dari Bapak) :");
14         objectAnak.show_variabel();
15     }
16 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Object Anak (Superclass dari Bapak) :
Nilai a = 0
Nilai b = 0
Nilai c = 5
PS D:\semester 4\praktikum2-inherentence\percobaan7>
```

Kemudian lakukan modifikasi pada method `show_variabel()` pada class `Anak`. Gunakan `super` untuk menampilkan nilai `a` dan `b` (memanfaatkan method yang sudah ada pada superclass).

```
J Bapak1.java > Anak
1 //Modifikasi pada Method show_variabel() di Class Anak
2
3 class Anak extends Bapak {
4     int c;
5
6     public Anak() {
7         a = 1;
8         b = 2;
9     }
10
11     void show_variabel() {
12         super.show_variabel();
13         System.out.println("Nilai c = " + c);
14     }
15 }
```

```
J InheritExample1.java > InheritExample1
1 public class InheritExample1 {
    Run | Debug
2     public static void main(String[] args) {
3         Bapak objectBapak = new Bapak();
4         Anak objectAnak = new Anak();
5
6         objectBapak.a = 1;
7         objectBapak.b = 2;
8
9         System.out.println(x:"Object Bapak (Superclass):");
10        objectBapak.show_variabel();
11
12        objectAnak.a = 3;
13        objectAnak.b = 4;
14        objectAnak.c = 5;
15    }
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Run: Inhe

```
Object Bapak (Superclass):
Nilai a = 1
Nilai b = 2
Object Anak (Subclass):
Nilai a = 3
Nilai b = 4
Nilai c = 5
PS D:\semester 4\praktikum2-inherentence\percobaan7>
```

Penjelasan :

- Keyword super digunakan untuk mengakses method show\_variabel() di class Bapak.
- Method show\_variabel() di class Bapak akan mencetak nilai a dan b.
- Setelah itu, kode di class Anak akan mencetak nilai c.

Artinya :

Modifikasi ini menunjukkan bagaimana super dapat digunakan untuk mengakses method di superclass, menghindari duplikasi kode, dan meningkatkan maintainability.

## PERCOBAAN 8

Percobaan berikut ini menunjukkan penggunaan overriding method pada kelas Parent dan subkelas Baby, saat dilakukan pemanggilan konstruktor superclass dengan menggunakan `super`

### Parent Class:

- a. Ini adalah sebuah class bernama "Parent".
- b. Ia memiliki variabel `parentName` untuk menyimpan nama orang tua (String).
- c. Ia memiliki dua konstruktor
  - Konstruktor tanpa parameter (`Parent()`) tidak melakukan apapun.
  - Konstruktor dengan parameter `parentName` (String) untuk menginisialisasi variabel `parentName`.
  - Kedua konstruktor mencetak pesan "Kontruktor parent" saat dijalankan.

### Baby Class:

- a. Ini adalah class bernama "Baby" yang mewarisi dari class "Parent". Artinya, "Baby" memiliki semua kemampuan yang dimiliki "Parent".
- b. Ia memiliki variabel tambahan `babyName` untuk menyimpan nama bayi (String).
- c. Ia memiliki konstruktor dengan parameter `babyName` (String)
  - Konstruktor ini:
  - Memanggil konstruktor tanpa parameter dari class "Parent" menggunakan `super()`. Ini memastikan inisialisasi awal dari "Parent" terlebih dahulu.
  - Menginisialisasi variabel `babyName`.
  - Mencetak pesan "Kontruktor Baby" dan nilai `babyName` saat dijalankan.
- d. Ia memiliki method `Cry()` yang mencetak pesan "Owek owek" saat dipanggil.

### Singkatnya :

- Class "Baby" mewarisi sifat dari class "Parent" (memiliki nama orang tua) dan menambahkan kemampuan khusus bayi (menangis).
- Kode ini menunjukkan konsep pewarisan (inheritance) dalam Java.

```
J Parent.java > Parent > Parent(String)
1 public class Parent {
2     String parentName;
3     Parent () {}
4     Parent (String parentName) {
5         this.parentName = parentName;
6         System.out.println(x:"Kontruktor parent");
7     }
8 }
9
10 class Baby extends Parent {
11     String babyName;
12
13     Baby (String babyName) {
14         super();
15         this.babyName = babyName;
16         System.out.println(x:"Kontruktor Baby");
17         System.out.println(babyName);
18     }
19
20     public void Cry() {
21         System.out.println(x:"Owek owek");
22     }
23 }
24
25
26
```

### Kesimpulan:

Kode ini membuat objek Baby bernama "Bayi Lucu" dan kemudian menyuruhnya menangis.

```
J Parent.java J Main.java X
J Main.java > Main
1 public class Main {
2     Run | Debug
3     public static void main(String[] args) {
4         Baby baby = new Baby(babyName:"Bayi Lucu");
5         baby.Cry();
6     }
7 }
8 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Run: Main
eStorage\4485d06573a365ed498b3903c3f5e48c\redhat.java\jdt_ws\percobaan8_fc949b99\bin' 'Main'
Kontruktor Baby
Bayi Lucu
Owek owek
PS D:\semester 4\praktikum2-inherentence\percobaan8>
```

