

## **Tugas Praktikum Sesi 3**

### **Analisa setiap percobaan pada file Praktikum**

**Moch Adriq Fadillah**

**20220040047**

#### **Percobaan 1**

Pada percobaan pertama menggunakan kata kunci super itu untuk memanggil atribut dari class parent nya yaitu nilai x pada kelas parent ialah 5

Berbeda jika tidak menggunakan apa apa dan langsung memanggil atribut x nya, itu akan otomatis memanggil nilai x yang ada di parameter

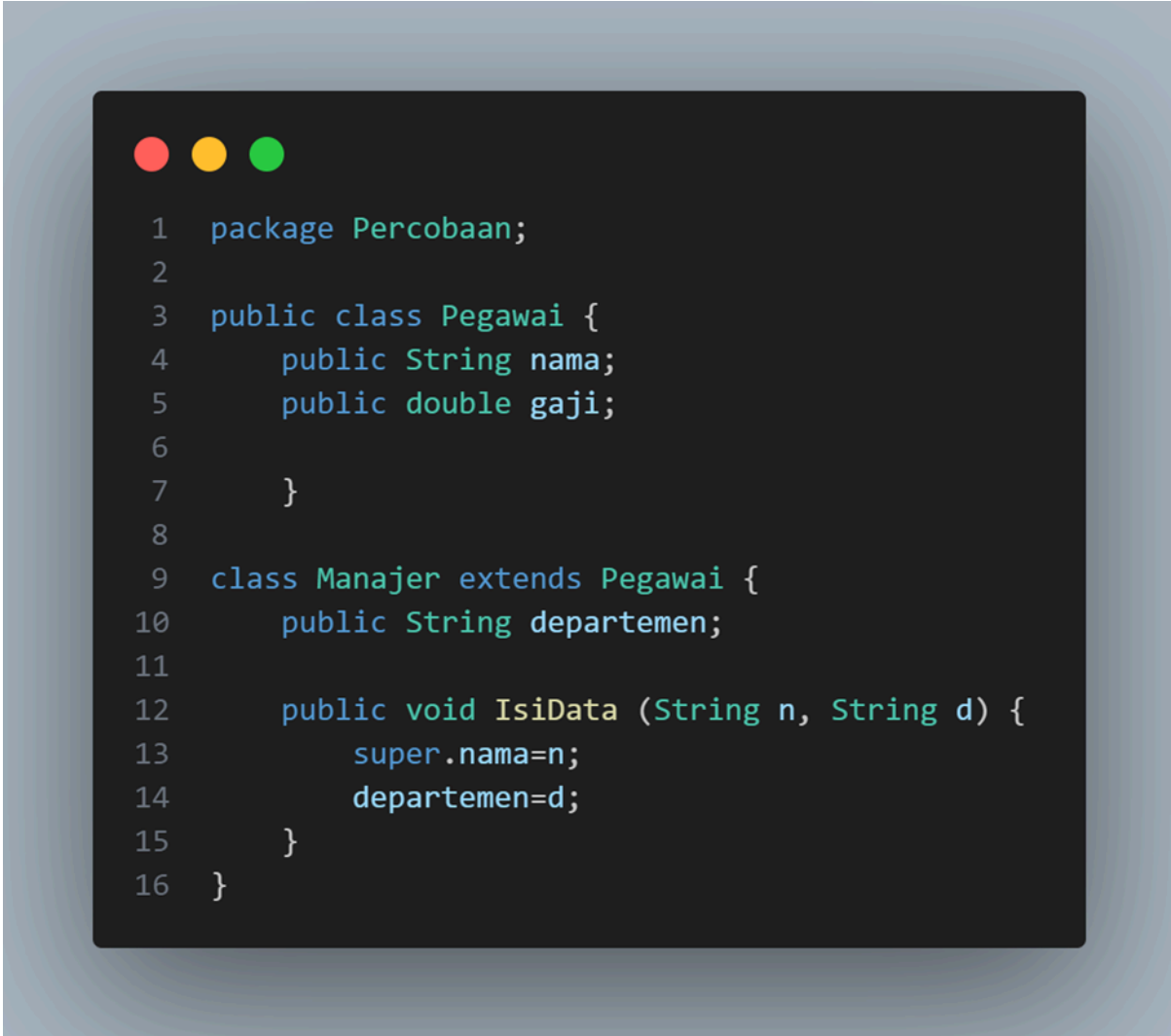
Berbeda juga jika menggunakan kata kunci this, itu akan memanggil atribut yang ada pada class nya sendiri.

```
Nilai x sebagai parameter = 20  
Data member x di class child = 10  
Data member x di class parent = 5
```

## Percobaan 2

Pada percobaan ke 2 menunjukkan penggunaan kontrol akses terhadap atribut parent Class, yaitu atribut String nama.

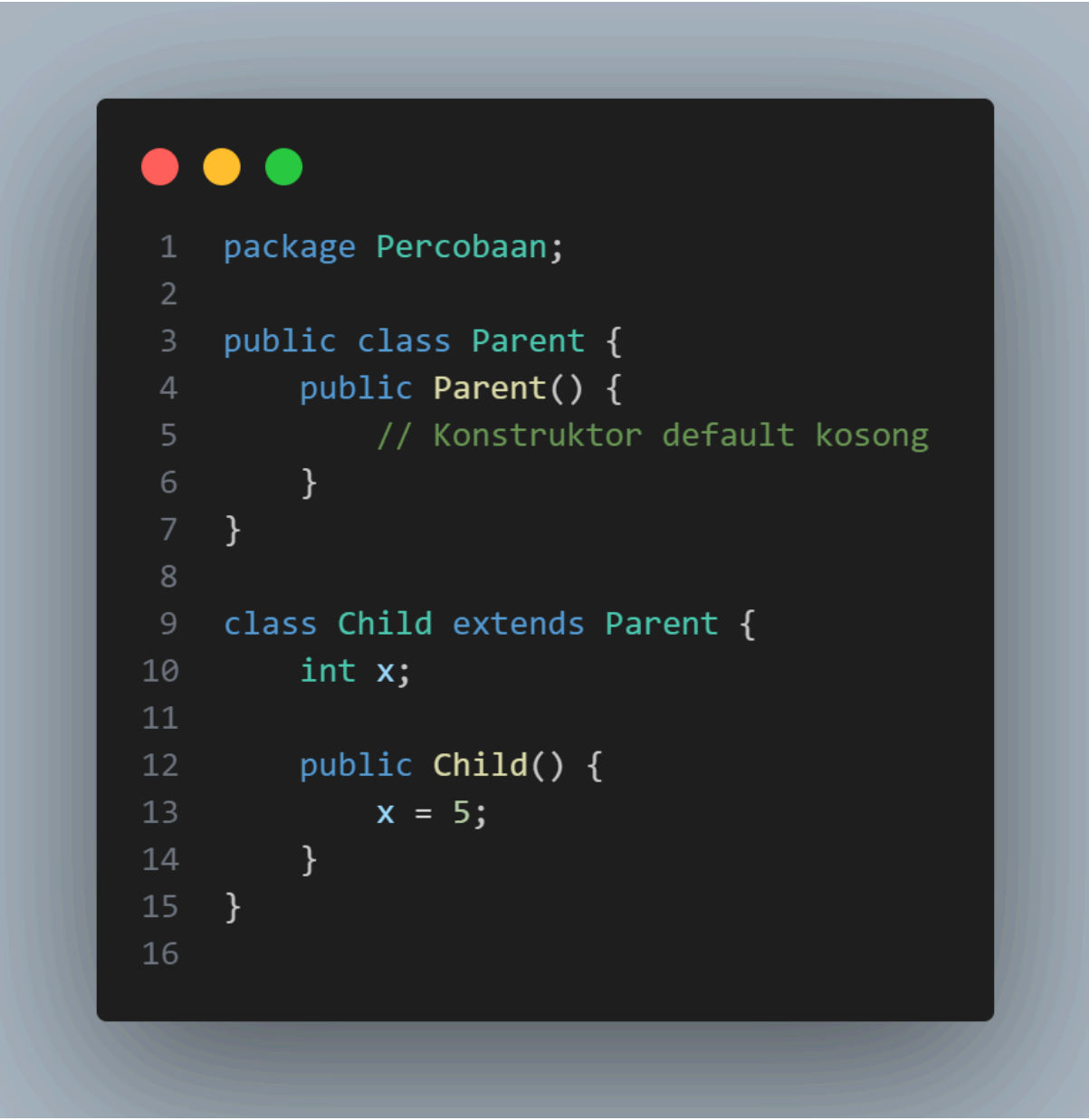
Itu disebabkan oleh Acces Control pada atribut nama ialah private yang dimana itu hanya bisa di akses pada class nya sendiri, oleh karena itu perlu nya ada perubahan acces control dari private menjadi public class. Ketika pada pemanggilan nilai nama juga diperlukan `super` untuk bisa memanggil atribut dari class parent nya.

A screenshot of a code editor with a dark background and light-colored text. The code is written in Java and shows a class hierarchy. The first class is 'Pegawai' with two public attributes: 'nama' (String) and 'gaji' (double). The second class is 'Manajer' which extends 'Pegawai' and has a public attribute 'departemen' (String). It also has a public method 'IsiData' that takes two String arguments, 'n' and 'd', and sets 'super.nama' to 'n' and 'departemen' to 'd'. The code is numbered from 1 to 16.

```
1 package Percobaan;
2
3 public class Pegawai {
4     public String nama;
5     public double gaji;
6
7 }
8
9 class Manajer extends Pegawai {
10     public String departemen;
11
12     public void IsiData (String n, String d) {
13         super.nama=n;
14         departemen=d;
15     }
16 }
```

### Percobaan 3

Kode tersebut menyebabkan kesalahan karena kelas `Child` mencoba untuk mengakses konstruktor default kelas induknya (`Parent`), namun kelas `Parent` tidak memiliki konstruktor default yang secara otomatis diwariskan. Solusinya adalah dengan menambahkan konstruktor default pada kelas `Parent` atau menambahkan konstruktor pada kelas `Child` yang secara eksplisit memanggil konstruktor kelas induk menggunakan kata kunci `super()`. Dengan demikian, Anda dapat memastikan bahwa konstruktor diwariskan dengan benar dari kelas induk. Misalnya, menambahkan konstruktor default kosong pada kelas `Parent` akan memungkinkan kelas `Child` untuk diinisialisasi tanpa kesalahan.




```
1  package Percobaan;
2
3  public class Parent {
4      public Parent() {
5          // Konstruktor default kosong
6      }
7  }
8
9  class Child extends Parent {
10     int x;
11
12     public Child() {
13         x = 5;
14     }
15 }
16
```

#### Percobaan 4

Pada program itu, hasil output nya yaitu seperti di bawah ini :

```
Name:John  
Salary:5000000.0  
Department:Financial  
Name:Michael  
Salary:15000.0  
Department:Accounting
```

Pada program tersebut terdapat satu kekurangan yaitu pada import yang harus meng import



```
1  import java.util.Date;
```

Kode tersebut mendefinisikan dua kelas, yaitu `Employee` dan `Manager`, serta sebuah kelas untuk menguji fungsionalitas keduanya yang disebut `TestManager`. Kelas `Employee` memiliki atribut `Name`, `Salary`, dan `birthDate`, yang masing-masing memiliki kontrol akses `private`. Ada empat konstruktor yang berbeda di kelas `Employee`, termasuk konstruktor default, serta getter untuk `Name` dan `Salary`. Kelas `Manager` adalah turunan dari `Employee` yang menambahkan atribut `department` dan memiliki tiga konstruktor yang berbeda, yang masing-masing menginisialisasi objek dengan parameter yang berbeda. `TestManager` digunakan untuk menguji pembuatan objek `Manager` dengan berbagai konstruktor, dan mencetak nilai-nilai atributnya menggunakan getter yang disediakan. Kode ini menunjukkan penggunaan konstruktor, kontrol akses, dan getter di Java untuk mengatur dan mengakses nilai-nilai objek kelas.

## Percobaan 5

Kode yang diberikan mendefinisikan tiga kelas utama: `MoodyObject`, `SadObject`, dan `HappyObject`, serta kelas pengujian yang disebut `MoodyTest`. Kelas `MoodyObject` memiliki beberapa metode dan atribut, termasuk `getMood()`, `speak()`, `laugh()`, dan `cry()`. Metode `getMood()` bertujuan untuk mengembalikan mood dari objek `MoodyObject`, yang dapat diubah dalam kelas turunannya. `speak()` digunakan untuk mencetak mood menggunakan metode `getMood()`. Dalam kelas turunannya, yaitu `SadObject` dan `HappyObject`, metode `getMood()` dioverride untuk mengembalikan mood yang sesuai dengan masing-masing kelas. Metode `cry()` di `SadObject` dan `laugh()` di `HappyObject` masing-masing digunakan untuk menampilkan tindakan yang sesuai dengan mood. Kelas `MoodyTest` digunakan untuk menguji fungsi dari objek-objek yang dibuat dari kelas-kelas tersebut. Dalam `main()` dari `MoodyTest`, objek `MoodyObject` pertama diinisialisasi dan metodenya diuji. Kemudian, objek diganti dengan objek `HappyObject` dan `SadObject` untuk menguji polimorfisme dalam pemanggilan metode, menunjukkan kemampuan untuk memanggil metode yang sesuai dengan jenis objeknya.

Output nya menjadi seperti di bawah :

```
I am moody  
I am happy  
Hahaha  
I am sad  
Hoo hoo
```

## Percobaan 6

Kode yang diberikan merupakan contoh implementasi kelas Java sederhana yang terdiri dari dua kelas, yaitu kelas `A` dan `B`, serta sebuah kelas `Main` yang berisi metode `main()` sebagai titik masuk program. Kelas `A` memiliki beberapa variabel instance yang memiliki kontrol akses default, yaitu `var\_a`, `var\_b`, `var\_c`, dan `var\_d`. Saat objek dari kelas `A` dibuat, konstruktor `A` dijalankan yang mencetak pesan "Konstruktor A dijalankan". Kelas `B` merupakan turunan dari kelas `A` dan memiliki konstruktor sendiri yang memodifikasi nilai dari variabel `var\_a` dan `var\_b`. Dalam metode `main()` dari kelas `Main`, objek dari kelas `A` dan `B` dibuat, dan nilai-nilai variabel di objek-objek tersebut dicetak untuk menunjukkan hasil dari konstruktor dan modifikasi yang dilakukan di dalam konstruktor kelas `B`.

```
Objek A dibuat
Konstruktor A dijalankan
Menampilkan nama variabel objek aa
Variabel A
Variabel B
Variabel C
Variabel D

Objek B dibuat
Konstruktor A dijalankan
Konstruktor B dijalankan
Menampilkan nama variabel objek bb
Var a dari class B
Var b dari class B
Variabel C
Variabel D
```

## Percobaan 7

Perhatikan bahwa dalam kode yang diberikan, kita melihat penggunaan variabel `a`, `b`, dan `c` dalam kelas `Bapak` dan `Anak`. Ketika objek dari kelas `Bapak` dan `Anak` dibuat, nilai-nilai untuk variabel-variabel ini diatur dengan menggunakan operator penugasan (`=`). Variabel `a` dan `b` diatur nilainya pada objek `objectBapak`, sedangkan variabel `c` diatur nilainya pada objek `objectAnak`. Kemudian, ketika metode `showVariabel()` dipanggil untuk setiap objek, nilai-nilai variabel `a`, `b`, dan `c` dicetak. Penting untuk dicatat bahwa nilai variabel `a` dan `b` pada objek `objectAnak` diwarisi dari kelas `Bapak`, sedangkan variabel `c` hanya ada di objek `objectAnak`, yang merupakan turunan dari `Bapak`. Ini menunjukkan bahwa variabel-variabel ini dapat dikelola secara terpisah di setiap objek, bahkan jika mereka memiliki nama yang sama dalam kelas-kelas yang berbeda.

output nya seperti di bawah ini ;

```
Object Bapak (Superclass):  
Nilai a=1  
Nilai b=1  
Object Anak (Subclass dari Bapak):  
Nilai a=0  
Nilai b=0  
Nilai c=5
```

## Percobaan 8

Kode tersebut menunjukkan dua kelas, yaitu `Parent` dan `Baby`. Kelas `Parent` memiliki satu variabel `parentName` yang digunakan untuk menyimpan nama orang tua. Kelas ini memiliki dua konstruktor, satu tanpa parameter dan satu dengan parameter `parentName`. Konstruktor tanpa parameter mencetak pesan "Konstruktor parent", sedangkan konstruktor dengan parameter akan menginisialisasi variabel `parentName` dengan nilai yang diberikan dan mencetak pesan yang sama. Kelas `Baby` adalah turunan dari kelas `Parent` dan memiliki tambahan variabel `babyName` untuk menyimpan nama bayi. Konstruktor kelas `Baby` menerima satu parameter `babyName`, mencetak pesan "Konstruktor Baby", menampilkan nilai `babyName`, dan menginisialisasi variabel `babyName`. Selain itu, kelas `Baby` memiliki metode `Cry()` yang mencetak respons bayi "Owek owek".