

RESPONSI PBO

NAMA : FIKRY MUMTAZ PRATAMA
NIM : H1D024106
SHIFT : SHIFT B
TANGGAL : 07/12/2025

JAWAB

Penjelasan Class Karyawan

1. Deklarasi Variabel

```
protected String nama;  
protected double gajiPokok;
```

Class *Karyawan* memiliki dua atribut dasar:

- **nama** → menyimpan nama pegawai
- **gajiPokok** → menyimpan gaji utama yang diterima setiap bulan

Modifier **protected** digunakan agar variabel ini dapat diakses oleh subclass (misalnya class *Manajer*), namun tetap tidak bersifat publik.

2. Constructor Karyawan

```
Karyawan(String nama, double gajiPokok) {  
    this.nama = nama;  
    this.gajiPokok = gajiPokok;  
}
```

Constructor ini bertugas menginisialisasi objek *Karyawan* dengan nilai:

- **nama**
- **gajiPokok**

Kata kunci **this** digunakan untuk membedakan antara variabel class dan parameter constructor.

3. Method **tampilInfo()**

```
void tampilInfo() {  
    System.out.println("Nama: " + nama + " | Gaji Pokok: Rp " + gajiPokok);  
}
```

Method ini mencetak informasi dasar seorang karyawan, yaitu nama dan gaji pokok. Method ini juga menjadi *method yang dapat dioverride* oleh subclass agar perilaku dapat dimodifikasi.

Penjelasan Class Manajer

1. Pewarisan (Inheritance)

```
class Manajer extends Karyawan
```

Class *Manajer* adalah turunan dari class *Karyawan*, sehingga seluruh atribut dan method parent dapat digunakan di dalam subclass.

2. Deklarasi Variabel Tambahan

```
double tunjangan;
```

Manajer memiliki variabel tambahan berupa:

- **tunjangan** → pendapatan ekstra di luar gaji pokok

Variabel ini hanya dimiliki oleh Manajer dan tidak ada pada Karyawan biasa.

3. Constructor Manajer

```
Manajer(String nama, double gajiPokok, double tunjangan) {  
    super(nama, gajiPokok);  
    this.tunjangan = tunjangan;  
}
```

Constructor Manajer:

- memanggil constructor parent **menggunakan super()**, karena Manajer memiliki nama dan gaji pokok seperti Karyawan pada umumnya
- mengisi nilai tunjangan yang hanya dimiliki oleh objek Manajer

Penggunaan **super()** merupakan cara resmi untuk mengakses constructor superclass dalam pewarisan.

4. Method Overriding **tampilInfo()**

```
@Override
```

```
void tampilInfo() {  
    System.out.println("Nama: " + nama +  
        " | Gaji Pokok: Rp " + gajiPokok +  
        " | Tunjangan: Rp " + tunjangan);
```

```
        System.out.println("Total Pendapatan: Rp " + (gajiPokok + tunjangan));  
    }
```

Di class Manajer, method `tampillInfo` di-*override* untuk:

- menampilkan data tambahan (tunjangan)
- menghitung total pendapatan (gaji pokok + tunjangan)
- menampilkan informasi yang lebih lengkap dibanding parent class

Keyword `@Override` digunakan sebagai penanda bahwa method ini menggantikan versi method yang ada di superclass.

Penjelasan Class UjiKaryawan

Class ini berfungsi sebagai *driver program* atau tempat pengujian objek dari class Karyawan dan Manajer.

1. Mencetak Header

```
System.out.println("==> DATA KARYAWAN TECHMAJU ==>");
```

Menampilkan judul program agar output lebih rapi dan informatif.

2. Membuat Objek Karyawan Biasa

```
Karyawan budi = new Karyawan("Budi Santoso", 4000000);
```

Objek ini dibuat menggunakan constructor Karyawan dengan:

- nama = Budi Santoso
- gaji pokok = 4.000.000

3. Menampilkan Informasi Karyawan Biasa

```
System.out.println("Status: Karyawan Biasa");
```

```
budi.tampillInfo();
```

Program menampilkan status karyawan dan memanggil method `tampillInfo` untuk mencetak data Budi.

4. Membuat Objek Manajer

```
Manajer siti = new Manajer("Siti Aminah", 6000000, 2500000);
```

Objek Manajer dibuat dengan informasi lengkap:

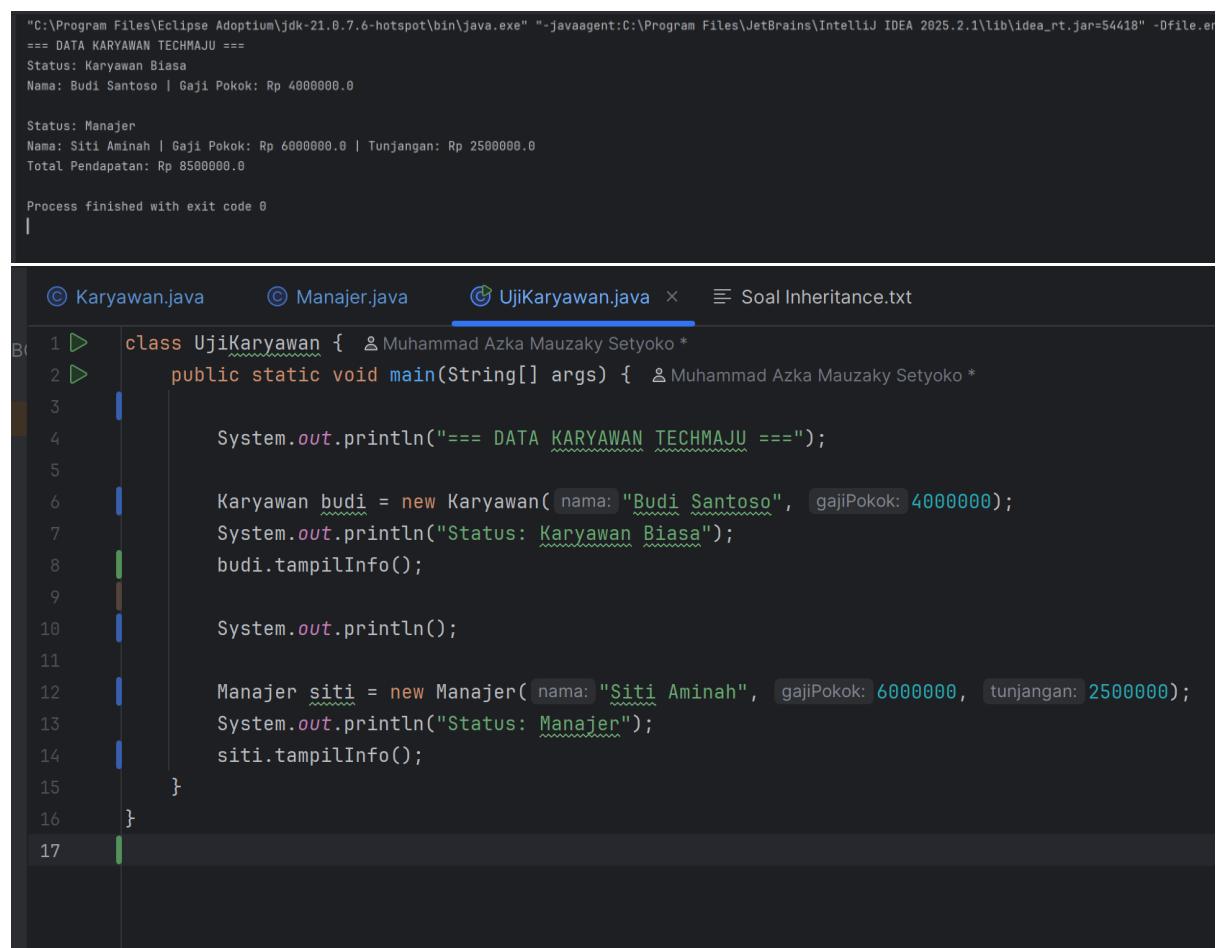
- nama = Siti Aminah
- gaji pokok = 6.000.000
- tunjangan = 2.500.000

Constructor Manajer akan memanggil constructor Karyawan melalui super().

5. Menampilkan Informasi Manajer

```
System.out.println("Status: Manajer");
siti.tampilInfo();
```

Output mencetak informasi lengkap Manajer, termasuk tunjangan dan total pendapatan.



The screenshot shows an IDE interface with two tabs open: "Karyawan.java" and "Manajer.java". The "UjiKaryawan.java" tab is currently active, containing the following code:

```
1 class UjiKaryawan {
2     public static void main(String[] args) {
3         System.out.println("== DATA KARYAWAN TECHMAJU ==");
4
5         Karyawan budi = new Karyawan( nama: "Budi Santoso", gajiPokok: 4000000 );
6         System.out.println("Status: Karyawan Biasa");
7         budi.tampilInfo();
8
9         System.out.println();
10
11        Manajer siti = new Manajer( nama: "Siti Aminah", gajiPokok: 6000000, tunjangan: 2500000 );
12        System.out.println("Status: Manajer");
13        siti.tampilInfo();
14    }
15 }
16
17 }
```

The output window above the code editor shows the execution results:

```
"C:\Program Files\Eclipse Adoptium\jdk-21.0.7.6-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.2.1\lib\idea_rt.jar=54418" -Dfile.encoding=UTF-8
== DATA KARYAWAN TECHMAJU ==
Status: Karyawan Biasa
Nama: Budi Santoso | Gaji Pokok: Rp 4000000.0

Status: Manajer
Nama: Siti Aminah | Gaji Pokok: Rp 6000000.0 | Tunjangan: Rp 2500000.0
Total Pendapatan: Rp 8500000.0

Process finished with exit code 0
```

The screenshot shows a Java code editor interface with two tabs open: `Manajer.java` and `Karyawan.java`. The `Manajer.java` tab is active.

`Manajer.java` content:

```
1 class Manajer extends Karyawan { 11 usages  ↗ Muhammad Azka Mauzaky Setyoko *
2     double tunjangan; 3 usages
3
4     Manajer(String nama, double gajiPokok, double tunjangan) { 10 usages new *
5         super(nama, gajiPokok);
6         this.tunjangan = tunjangan;
7     }
8
9     @Override 2 usages  ↗ Muhammad Azka Mauzaky Setyoko *
10    void tampilInfo() {
11        System.out.println("Nama: " + nama +
12            " | Gaji Pokok: Rp " + gajiPokok +
13            " | Tunjangan: Rp " + tunjangan);
14        System.out.println("Total Pendapatan: Rp " + (gajiPokok + tunjangan));
15    }
16
17 }
```

`Karyawan.java` content:

```
1 class Karyawan { 13 usages 1 inheritor  ↗ Muhammad Azka Mauzaky Setyoko *
2     protected String nama; 3 usages
3     protected double gajiPokok; 4 usages
4
5     Karyawan(String nama, double gajiPokok) { 12 usages new *
6         this.nama = nama;
7         this.gajiPokok = gajiPokok;
8     }
9
10    void tampilInfo() { 2 usages 1 override  ↗ Muhammad Azka Mauzaky Setyoko *
11        System.out.println("Nama: " + nama + " | Gaji Pokok: Rp " + gajiPokok);
12    }
13
14 }
```