

Nama : Mosyarofah

Nim : 202412032

Matkul : pemrograman berorientasi objek

## 1. Source code

```

1 class Karyawan:
2     # Constructor (__init__) untuk menginisialisasi atribut karyawan
3     def __init__(self, nama, id_karyawan, gaji_pokok):
4         # self.nama digunakan untuk menyimpan nama karyawan
5         self.nama = nama
6         # self.id_karyawan menyimpan ID unik dari karyawan
7         self.id_karyawan = id_karyawan
8         # self.gaji_pokok menyimpan nilai gaji pokok karyawan
9         self.gaji_pokok = gaji_pokok
10
11     # Method untuk menghitung gaji karyawan (default hanya gaji pokok)
12     def hitung_gaji(self):
13         return self.gaji_pokok
14
15     # Method untuk menampilkan informasi dasar karyawan
16     def info(self):
17         # f-string digunakan agar mudah menampilkan variabel di dalam teks
18         return f"[{self.nama}], ID: [{self.id_karyawan}], Gaji: [{self.hitung_gaji()}]"
19
20
21 class Manager(Karyawan):
22     # Constructor Manager menambah atribut baru yaitu tunjangan
23     def __init__(self, nama, id_karyawan, gaji_pokok, tunjangan):
24         # Menanggil constructor milik parent class (Karyawan)
25         super().__init__(nama, id_karyawan, gaji_pokok)
26         # Menambahkan atribut baru: tunjangan
27         self.tunjangan = tunjangan
28
29     # Method ini "meng-override" (menimpa) method dari parent class
30     # Tujuannya agar perhitungan gaji menambahkan tunjangan
31     def hitung_gaji(self):
32         return self.gaji_pokok + self.tunjangan
33
34     # Override juga method info() agar menampilkan teks "Manager"
35     def info(self):
36         return f"Manager : [{self.nama}], ID: [{self.id_karyawan}], Gaji: [{self.hitung_gaji()}]"
37
38
39 class Programmer(Karyawan):
40     # Constructor Programmer menambah atribut baru yaitu bonus
41     def __init__(self, nama, id_karyawan, gaji_pokok, bonus):
42         # Menanggil constructor dari parent class (Karyawan)
43         super().__init__(nama, id_karyawan, gaji_pokok)
44         # Menambahkan atribut baru: bonus
45         self.bonus = bonus
46
47     # Override method hitung_gaji untuk menambah bonus ke gaji pokok
48     def hitung_gaji(self):
49         return self.gaji_pokok + self.bonus
50
51     # Override method info() agar menampilkan teks "Programmer"
52     def info(self):
53         return f"Programmer : [{self.nama}], ID: [{self.id_karyawan}], Gaji: [{self.hitung_gaji()}]"
54
55
56 # =====
57 # Bagian utama program (Main Program)
58 # =====
59 if __name__ == "__main__":
60     # Membuat objek dari class Manager dengan nama, ID, gaji pokok, dan tunjangan
61     manager1 = Manager("Mosyarofah", "MP01", 12000000, 5000000)
62
63     # Membuat objek dari class Programmer dengan nama, ID, gaji pokok, dan bonus
64     programmer1 = Programmer("Sarah", "P001", 10000000, 2000000)
65
66     # Menanggil method info() dari masing-masing objek
67     # Karena method info() di-override, hasilnya akan berbeda tiap class
68     print(manager1.info())    # Menampilkan info Manager
69     print(programmer1.info()) # Menampilkan info Programmer
70

```

## 2. Output



```

PS C:\Users\Asus-PC\OneDrive\Documents\P80 semester 3> & C:/Users/Asus-PC/AppData/Local/Programs/Python/Python39-32/Python.exe C:\Users\Asus-PC\OneDrive\Documents\P80 semester 3\Manajemen_karyawan.py
Manager : MosyaroFah, ID: M001, Gaji: 15000000
Programmer : Sarah, ID: P001, Gaji: 12000000
PS C:\Users\Asus-PC\OneDrive\Documents\P80 semester 3>

```

**Penjelasan :**

Kode “Sistem Manajemen Karyawan” ini dibuat untuk menerapkan konsep Object Oriented Programming (OOP) dalam bentuk inheritance dan polymorphism. Tujuan utamanya adalah untuk mengelola data dan perhitungan gaji karyawan secara terstruktur menggunakan pendekatan berorientasi objek. Class Karyawan berperan sebagai class induk yang menyimpan atribut dasar seperti nama, ID, dan gaji pokok. Sementara itu, class Manager dan Programmer merupakan class turunan yang menambahkan atribut khusus seperti tunjangan dan bonus, serta melakukan method overriding untuk menyesuaikan perhitungan gaji sesuai dengan jabatan masing-masing.

Fungsi utama dari program ini adalah menghitung dan menampilkan gaji total karyawan berdasarkan jenis jabatannya. Program ini juga mempermudah proses pengelolaan data karena setiap jenis karyawan memiliki class dan metode yang spesifik namun tetap menggunakan struktur dasar yang sama dari class induk. Dengan begitu, jika suatu saat ada jenis karyawan baru atau sistem penggajian yang berbeda, programmer hanya perlu menambahkan subclass baru tanpa harus mengubah keseluruhan kode yang sudah ada.

Dalam kehidupan nyata, program seperti ini dapat diterapkan dalam sistem manajemen sumber daya manusia (HRD) di perusahaan. Misalnya, digunakan untuk menghitung gaji otomatis setiap karyawan, mengelompokkan karyawan berdasarkan jabatan, atau menghasilkan laporan penggajian bulanan. Dengan pendekatan OOP seperti ini, sistem menjadi lebih efisien, fleksibel, dan mudah dikembangkan sesuai kebutuhan perusahaan.