NAMA : RIZKY PUSPITA INDRASWARI

NIM : 242410103037

KELAS : PBO D

* **Progam**

using System;

class Karyawan

{

private string nama;

private string id;

private double gajiPokok;

public string Nama

{

get => nama;

set => nama = value;

}

public string ID

{

get => id;

set => id = value;

}

public double GajiPokok

{

get => gajiPokok;

set => gajiPokok = value;

}

public virtual double HitungGaji()

{

return gajiPokok;

}

}

class KaryawanTetap : Karyawan

{

private double bonusTetap = 500000;

public override double HitungGaji()

{

return GajiPokok + bonusTetap;

}

}

class KaryawanKontrak : Karyawan

{

private double potonganKontrak = 200000;

public override double HitungGaji()

{

return GajiPokok - potonganKontrak;

}

}

class KaryawanMagang : Karyawan

{

public override double HitungGaji()

{

return GajiPokok;

}

}

class Program

{

static void Main()

{

Console.Write("Masukkan jenis karyawan (tetap/kontrak/magang): ");

string jenis = Console.ReadLine().ToLower();

Console.Write("Masukkan nama: ");

string nama = Console.ReadLine();

Console.Write("Masukkan ID: ");

string id = Console.ReadLine();

Console.Write("Masukkan gaji pokok: ");

double gajiPokok = Convert.ToDouble(Console.ReadLine());

Karyawan karyawan;

switch (jenis)

{

case "tetap":

karyawan = new KaryawanTetap();

break;

case "kontrak":

karyawan = new KaryawanKontrak();

break;

case "magang":

karyawan = new KaryawanMagang();

break;

default:

Console.WriteLine("Jenis karyawan tidak valid.");

return;

}

karyawan.Nama = nama;

karyawan.ID = id;

karyawan.GajiPokok = gajiPokok;

Console.WriteLine("\nDetail Karyawan:");

Console.WriteLine($"Nama : {karyawan.Nama}");

Console.WriteLine($"ID : {karyawan.ID}");

Console.WriteLine($"Gaji Pokok : {karyawan.GajiPokok}");

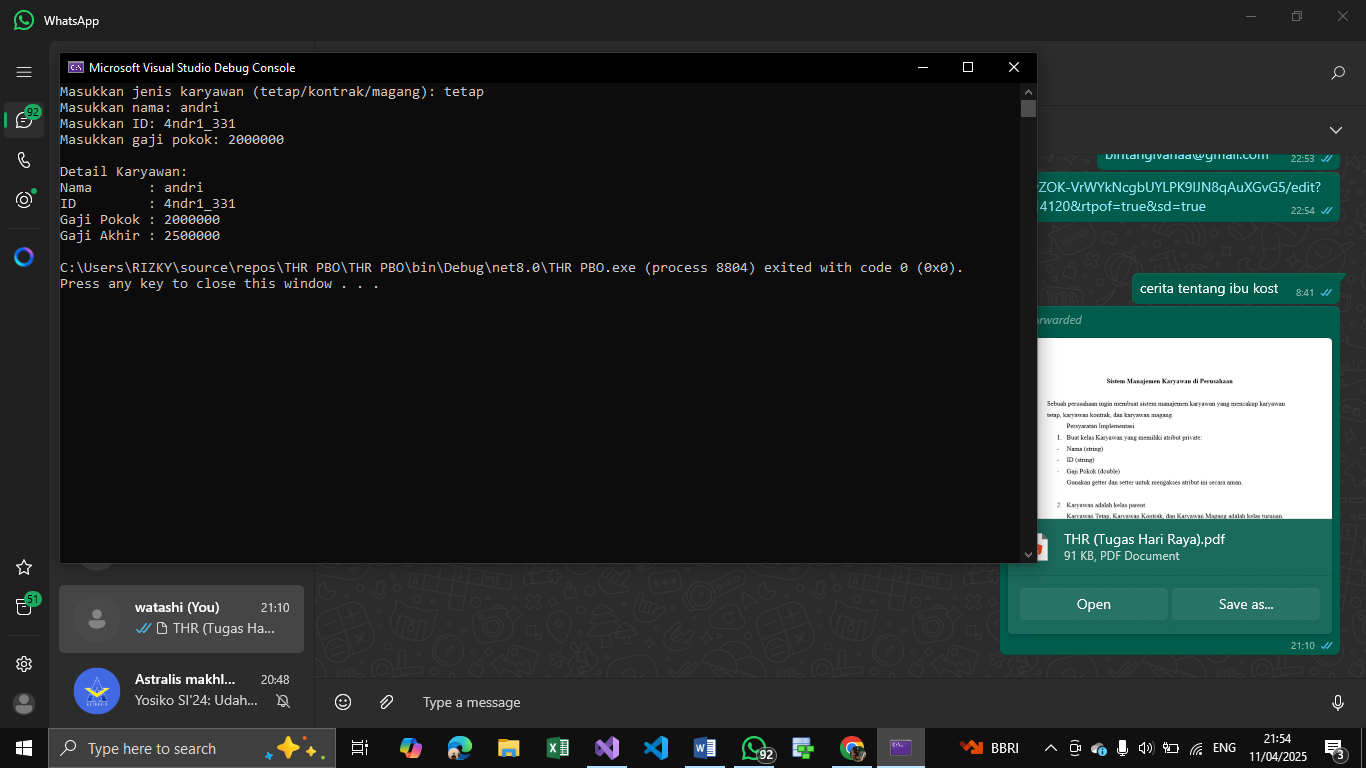
Console.WriteLine($"Gaji Akhir : {karyawan.HitungGaji()}");

}

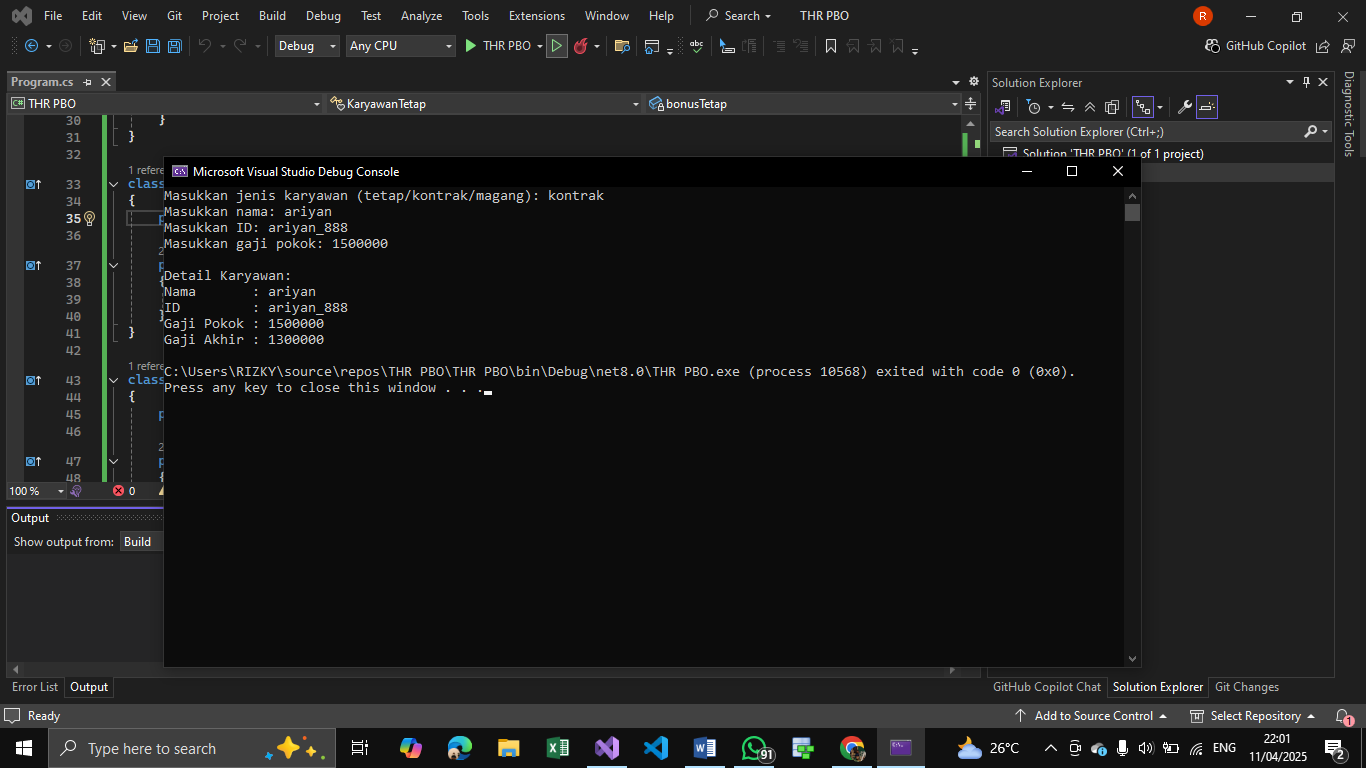
}

* **Hasil**

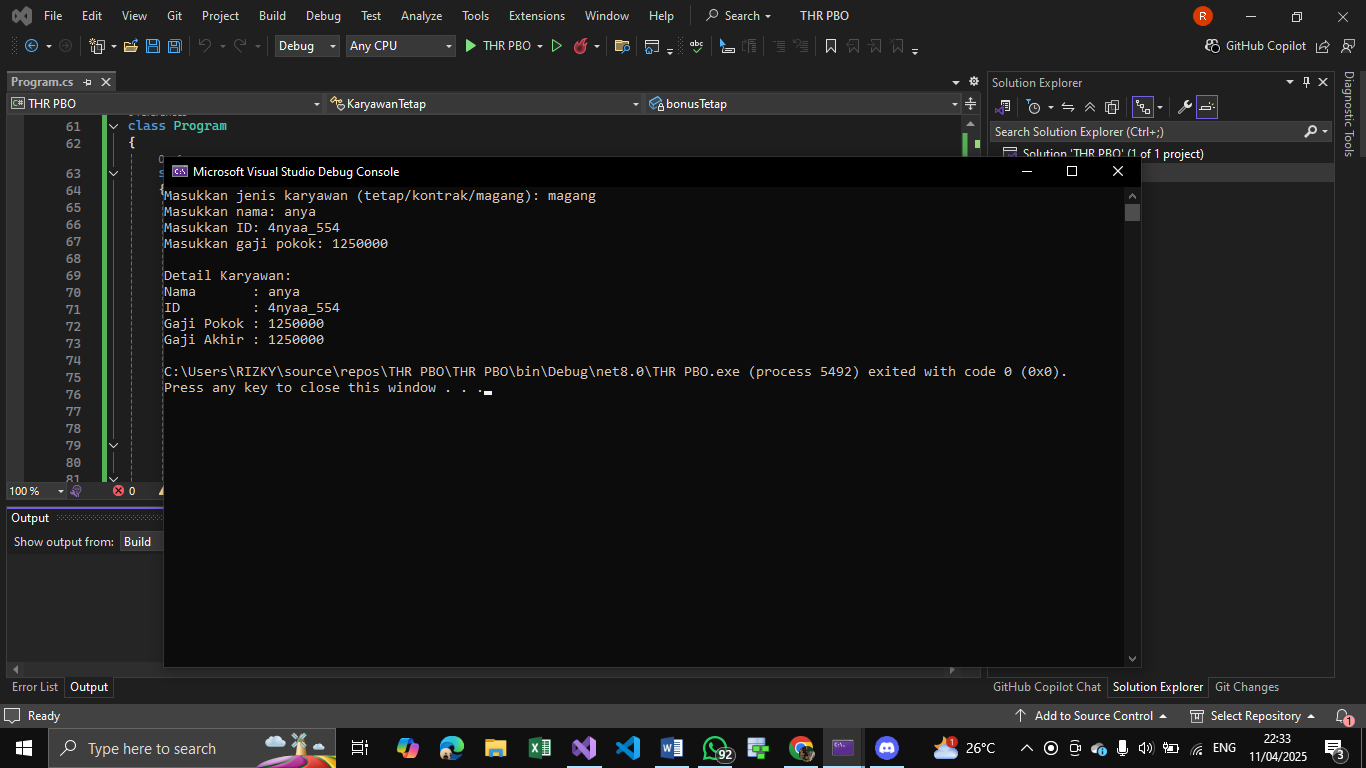
1. **Karyawan tetap**

****

1. **Karyawan kontrak**

****

1. **Karyawan magang**

****

* **Class diagram**

+--------------------------+

| Karyawan |

+--------------------------+

| - nama: string |

| - id: string |

| - gajiPokok: double |

+--------------------------+

| +Nama: string |

| +ID: string |

| +GajiPokok: double |

| +HitungGaji(): double|

+------------^-------------+

|

-----------------------------------------------------

| | |

+--------------------+ +------------------------+ +------------------------+

| KaryawanTetap | | KaryawanKontrak | | KaryawanMagang |

+--------------------+ +------------------------+ +------------------------+

| - bonusTetap | | - potonganKontrak | | |

+--------------------+ +------------------------+ +------------------------+

| +HitungGaji(): | | +HitungGaji(): | | +HitungGaji(): |

| double | | double | | double |

+--------------------+ +------------------------+ +------------------------+

Penjelasan class diagram :

1. Kelas Karyawan (Superclass)

Merupakan kelas induk dari semua jenis karyawan dengan atribut:

* **nama**: nama karyawan.
* **id**: ID karyawan.
* **gajiPokok**: gaji dasar setiap karyawan.

Metode:

* Getter dan Setter untuk atribut.
* **HitungGaji()**: metode virtual, yang akan dioverride oleh kelas turunan.

2. Kelas KaryawanTetap (Subclass)

Mewarisi dari **Karyawan**.

Atribut:

* **bonusTetap**: bonus tetap sebesar 500.000.

Override metode **HitungGaji()** → menghitung gaji dengan penambahan bonus.

3. Kelas **KaryawanKontrak** (Subclass)

Mewarisi dari **Karyawan.**

Atribut:

* **potonganKontrak**: potongan sebesar 200.000.

Override metode **HitungGaji()** → menghitung gaji dengan pengurangan potongan.

4. Kelas **KaryawanMagang** (Subclass)

Mewarisi dari **Karyawan.**

Tidak punya atribut khusus.

Override **HitungGaji()** → gaji hanya sesuai gaji pokok, tanpa potongan/bonus.

Relasi Antar Kelas

* Inheritance (pewarisan) ditunjukkan oleh panah segitiga dari subclass ke superclass (**^**).
* Semua subclass mewarisi atribut dan metode dari **Karyawan**, lalu mengubah (**override**) perilaku metode **HitungGaji()** sesuai kebutuhan masing-masing.
* **Penjelasan**

Program menggunakan 4 kelas:

1. **Karyawan**: kelas induk dengan atribut nama, id, dan gaji pokok. Memiliki metode HitungGaji() virtual.
2. **KaryawanTetap**: turunan Karyawan, memiliki bonus tetap Rp500.000.
3. **KaryawanKontrak**: turunan Karyawan, memiliki potongan Rp200.000.
4. **KaryawanMagang**: turunan Karyawan, tidak mendapat bonus atau potongan.

Program ini dirancang untuk mengelola data karyawan di sebuah perusahaan dengan menerapkan prinsip-prinsip **Object-Oriented Programming (OOP)** dalam bahasa C#. Tujuan utama dari program ini adalah memisahkan perhitungan gaji berdasarkan jenis karyawan, yaitu karyawan tetap, kontrak, dan magang. Dengan menggunakan konsep pewarisan (inheritance), kita bisa membuat satu kelas induk bernama **Karyawan** yang menyimpan atribut umum seperti nama, ID, dan gaji pokok. Kelas ini juga memiliki metode virtual **HitungGaji()** yang nantinya akan disesuaikan di setiap kelas turunan. Pendekatan ini membuat struktur program menjadi lebih rapi dan mudah dikembangkan.

Pada bagian kelas turunan, masing-masing jenis karyawan memiliki cara perhitungan gaji yang berbeda. Kelas **KaryawanTetap** menambahkan bonus tetap sebesar Rp500.000 pada gaji pokok. Sebaliknya, **KaryawanKontrak** memiliki potongan kontrak sebesar Rp200.000 yang mengurangi gaji pokok. Sedangkan pada **KaryawanMagang**, gaji akhir sama persis dengan gaji pokok tanpa tambahan atau potongan apa pun. Seluruh kelas ini mewarisi atribut dan metode dari kelas **Karyawan**, serta mengoverride metode **HitungGaji()** sesuai kebijakan masing-masing.

Proses eksekusi program berada pada metode **Main()**, yang dimulai dengan meminta input dari pengguna. Pengguna diminta memilih jenis karyawan dan memasukkan data seperti nama, ID, dan gaji pokok. Berdasarkan jenis karyawan yang dimasukkan, program kemudian menentukan kelas turunan mana yang akan digunakan untuk membuat objek karyawan. Setelah objek dibuat, data pengguna dimasukkan melalui property setter, lalu program menghitung gaji akhir dengan memanggil metode **HitungGaji()** yang sesuai dengan objek tersebut.

Setelah proses perhitungan selesai, program akan menampilkan hasil berupa detail karyawan seperti nama, ID, gaji pokok, dan gaji akhir. Karena metode **HitungGaji()** bersifat polymorphic, maka meskipun objeknya disimpan dalam variabel bertipe **Karyawan**, metode yang dipanggil akan tetap sesuai dengan kelas sebenarnya dari objek tersebut. Ini adalah contoh nyata dari penerapan konsep **polimorfisme** dalam OOP, di mana satu metode bisa berperilaku berbeda tergantung pada objek yang memanggilnya.

Secara keseluruhan, program ini tidak hanya berguna untuk menyederhanakan proses manajemen karyawan, tetapi juga memperlihatkan bagaimana konsep OOP dapat diterapkan untuk membuat program yang fleksibel, terstruktur, dan efisien. Jika suatu saat perusahaan ingin menambahkan jenis karyawan baru, kita cukup membuat kelas baru yang mewarisi dari Karyawan dan mengimplementasikan metode **HitungGaji()** sesuai kebutuhan, tanpa harus mengubah struktur program yang sudah ada.