Pemrograman Berorientasi Objek Praktik

~ ~ Meet 07 ~ ~

Program Studi Informatika
Universitas Teknologi Yogyakarta

Pewarisan
(*Inheritance*)

Objectives

- Menjelaskan konsep dasar pewarisan (inheritance) dalam OOP.
- Mengidentifikasi manfaat penggunaan pewarisan dalam perancangan program.
- Mengimplementasikan pewarisan tunggal dan berganda dalam Python.
- Menjelaskan hubungan antara pewarisan dan access modifier.

Review Singkat

- Minggu lalu: kita membahas kelas, objek, dan access modifier (public, protected, private).
- Setiap kelas dapat memiliki atribut dan method yang hanya dimiliki oleh objeknya.
- Tetapi, bagaimana jika kita ingin mewarisi atribut dan method tersebut ke kelas lain?

Apa Itu Pewarisan (Inheritance)?

- Pewarisan adalah mekanisme di OOP di mana sebuah kelas baru dapat mewarisi atribut dan method dari kelas lain.
- Kelas yang diwarisi disebut Superclass / Parent class.
- Kelas yang mewarisi disebut Subclass / Child class.
- Tujuannya: Reusability (kode dapat digunakan kembali).

Analogi Pewarisan

Misalnya:

- Kelas Hewan memiliki method makan().
- Kelas Kucing dan Anjing adalah turunan dari Hewan.
- Maka Kucing dan Anjing mewarisi kemampuan makan() dari Hewan.

Kode tidak perlu ditulis ulang — cukup diwarisi.

Sintaks Pewarisan di Python

```
class Parent:
    def __init__(self, nama):
        self.nama = nama

    def sapa(self):
        print(f"Halo, saya {self.nama}")

class Child(Parent):  # Child mewarisi Parent
    def main(self):
        print(f"{self.nama} sedang bermain")

obj = Child("Budi")
obj.sapa()   # diwarisi dari Parent
obj.main()   # milik Child
```

Output Program

```
Halo, saya Budi
Budi sedang bermain
```

Penjelasan:

- Child mewarisi sapa() dari Parent.
- Child juga bisa memiliki method sendiri: main().

Keyword super()

- Digunakan untuk memanggil konstruktor atau method dari kelas induk (Parent).
- Berguna ketika Subclass menambahkan inisialisasi tambahan.

Keyword super()

```
class Hewan:
    def __init__(self, nama):
        self.nama = nama

class Kucing(Hewan):
    def __init__(self, nama, warna):
        super().__init__(nama)
        self.warna = warna

    def info(self):
        print(f"{self.nama} berwarna {self.warna}")

k = Kucing("Milo", "putih")
k.info()
```

Keyword super()

- Output:

```
Milo berwarna putih
```

Mengapa Gunakan super()

- Tanpa super(), kita perlu memanggil konstruktor induk secara manual:

```
Hewan.__init__(self, nama)
```

- Tapi super() lebih fleksibel dan mendukung multiple inheritance.

Pewarisan vs Komposisi

- Pewarisan (is-a relationship):
  - Kucing adalah Hewan.
- Komposisi (has-a relationship):
  - Mobil memiliki Mesin.
- Kadang kita bisa memilih antara pewarisan dan komposisi tergantung konteks.

Overriding (Menimpa Method)

- Subclass dapat menimpa (override) method dari Parent agar perilakunya berbeda.

```
class Hewan:
    def suara(self):
        print("Hewan bersuara")

class Kucing(Hewan):
    def suara(self):
        print("Meow!")

Kucing().suara()   # Output: Meow!
```

Manfaat Overriding

- Memberikan perilaku spesifik pada subclass.
- Menghindari duplikasi kode.
- Menunjukkan polimorfisme dalam OOP (objek berbeda bisa merespon dengan cara berbeda).

Multi-level Inheritance

- Sebuah kelas dapat mewarisi kelas lain yang juga merupakan hasil pewarisan.

```
class A: pass
class B(A): pass
class C(B): pass
```

- C mewarisi dari B, dan B mewarisi dari A.

Multi-level Inheritance

- Python mendukung pewarisan dari lebih dari satu parent class.

```
class A:
    def tampilA(self): print("Dari A")

class B:
    def tampilB(self): print("Dari B")

class C(A, B):
    pass

obj = C()
obj.tampilA()
obj.tampilB()
```

Urutan Pewarisan (MRO - Method Resolution Order)

- Ketika ada dua kelas induk, Python mengikuti urutan MRO (Method Resolution Order).
- Urutan diperiksa dari kiri ke kanan sesuai deklarasi pewarisan.Gunakan:

```
C.__mro__
```

Protected Member & Pewarisan

- Protected (_atribut) dapat diakses oleh subclass:

```
class Parent:
    def __init__(self):
        self._nilai = 100

class Child(Parent):
    def show(self):
        print(self._nilai)
```

Private Member & Pewarisan

- Private (__atribut) tidak dapat langsung diakses oleh subclass.

```
class Parent:
    def __init__(self):
        self.__saldo = 5000

class Child(Parent):
    def lihat(self):
        print(self.__saldo) # Error!
```

Solusi Mengakses Private

- Gunakan method getter atau setter di parent:

```
class Parent:
    def __init__(self):
        self.__saldo = 5000

    def get_saldo(self):
        return self.__saldo
```

Pewarisan di Dunia Nyata

- Pegawai → PegawaiTetap, PegawaiKontrak
- Kendaraan → Mobil, Motor
- Mahasiswa → MahasiswaAktif, MahasiswaAlumni

Kesalahan Umum Saat Menggunakan Inheritance

- Lupa memanggil super().__init__().
- Menimpa method tanpa sengaja (typo nama method).
- Terlalu dalam hierarki pewarisan → sulit dipelihara.

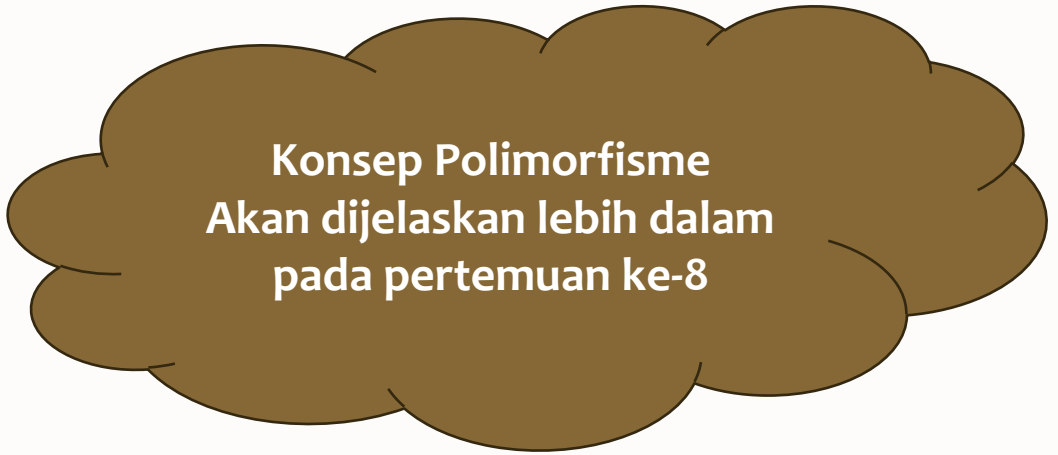
Kapan Sebaiknya Gunakan Pewarisan

Gunakan jika:

- Ada hubungan logis "is-a".
- Kelas turunan memang perlu berbagi perilaku induknya.
- Hindari jika hanya untuk menghemat baris kode.

Pewarisan vs Polimorfisme

- Inheritance → mewarisi struktur & perilaku.
- Polymorphism → objek turunan dapat berperilaku berbeda meski punya method yang sama.



Manfaat Pewarisan dalam Proyek Besar

- Kode modular dan terorganisir.
- Mudah diperluas tanpa ubah kelas dasar.
- Reusabilitas tinggi → efisiensi pengembangan.

Penutup

- Pewarisan: hubungan is-a antar kelas.
- Subclass dapat mewarisi dan menimpa perilaku Parent.
- super() → memanggil konstruktor induk.
- Gunakan inheritance dengan bijak untuk efisiensi desain.

Latihan Di Kelas

Latihan I

Instruksi:Buat program dengan struktur:

- class Akun (memiliki atribut saldo dan method setor, tarik)
- class AkunReward (turunan Akun, menambah sistem poin setiap kali setor)
- Tampilkan hasil akhir saldo dan poin.

Contoh Hasil Program

```
Setor: 1000 → saldo = 1000, poin = 10
Setor: 500 → saldo = 1500, poin = 15
```

Latihan I

- class Akun
  - Atribut: saldo
  - Method: setor(jumlah), tarik(jumlah)
- class AkunReward(Akun)
  - Turunan dari Akun.
  - Menambahkan atribut baru: poin.
  - Overriding method setor() dan tarik() untuk menghitung reward.

Latihan I

- Aturan Reward

Jenis Transaksi	Kondisi	Poin Reward yang Diperoleh
Setor Tunai	jumlah ≤ 1000	+1 poin per transaksi
Setor Tunai	1001 ≤ jumlah ≤ 5000	+5 poin
Setor Tunai	jumlah > 5000	+10 poin
Tarik Tunai	jumlah ≤ 1000	-1 poin
Tarik Tunai	jumlah > 1000	-3 poin

Latihan II

- Buat kelas Karyawan, lalu turunkan menjadi Manajer dan Staf Khusus.
- Setiap kelas punya cara berbeda dalam menghitung bonus.
- Terapkan overriding method hitung_bonus().

Latihan II

- Karyawan (superclass) — kelas umum yang menyimpan atribut dasar.
- Manajer (subclass) — turunan dari Karyawan dengan tambahan atribut jumlah_tim.
- Staf Khusus (subclass) — turunan dari Karyawan dengan tambahan atribut jumlah_kehadiran.

Latihan II

- Tabel Gaji Pokok

Jabatan	Gaji Pokok (Rp)	Keterangan
Karyawan (Umum)	5.000.000	Semua pegawai minimal mendapatkan gaji pokok ini.
Manajer	8.000.000	Tanggung jawab lebih besar karena memimpin tim.
Staf Khusus	4.000.000	Mendapat tambahan bonus berdasarkan kehadiran.

Latihan II

- Rumus Perhitungan Bonus

Jabatan	Rumus Bonus	Keterangan
Karyawan (Umum)	bonus = 0.05 × gaji_pokok	Semua pegawai mendapat bonus dasar 5%.
Manajer	bonus = (0.1 × gaji_pokok) + (0.02 × jumlah_tim × gaji_pokok)	Bonus 10% + tambahan 2% × jumlah anggota tim.
Staf Khusus	bonus = (0.05 × gaji_pokok) + (0.01 × jumlah_kehadiran × gaji_pokok)	Bonus 5% + tambahan 1% × jumlah kehadiran.

Tugas Di Rumah

Latihan di Rumah

- Buatlah sistem sederhana bernama Sistem Penjualan Produk menggunakan bahasa Python.
- Sistem ini memiliki kelas induk bernama Produk yang menyimpan informasi dasar semua barang.
- Terdapat kelas turunan untuk setiap kategori produk:
  - Elektronik
  - Pakaian
  - Makanan
- Setiap kategori memiliki aturan diskon berbeda, tergantung jenis produk dan harga total pembelian.

Latihan di Rumah - Kebutuhan Sistem

- Kelas Produk (superclass) memiliki atribut:
  - nama
  - Harga
  - Stok
- Setiap kelas turunan (Elektronik, Pakaian, Makanan) mewarisi dari Produk dan menambahkan:
  - atribut khusus (misalnya: garansi, ukuran, masa_kadaluarsa) ***"mhs bisa menambahkan atribut lain yang lain dengan teman-temannya"***
  - method hitung_diskon() yang memiliki aturan berbeda.
- Diskon dihitung berdasarkan kategori dan total harga pembelian.

Latihan di Rumah - Aturan Diskon Tiap Kategori

Kategori	Kondisi Pembelian (Rp)	Diskon (%)	Keterangan Tambahan
Elektronik	> 10.000.000	10%	Barang elektronik mahal, diskon kecil
	≤ 10.000.000	5%	Diskon tetap untuk harga di bawah 10 juta
Pakaian	> 1.000.000	15%	Diskon besar untuk pembelian banyak
	≤ 1.000.000	7%	Diskon sedang untuk pembelian kecil
Makanan	> 500.000	8%	Diskon berlaku jika pembelian banyak
	≤ 500.000	3%	Diskon ringan untuk pembelian kecil

Latihan di Rumah – Rumus Perhitungan Diskon

$$\text{Harga Akhir} = \text{Harga Barang} - (\text{Harga Barang} \times \text{Persentase Diskon})$$

Terima Kasih