LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 4

2411102441259

Malik Sabarullah Akbar

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : itsmalik57@gmail.com

Inheritance

Inheritance adalah salah satu metode pilar OOP yang sering digunakan untuk mewarisi parent class ke child class. Inheritance atau pewarisan bisa di analogikan seperti keluarga yang memiliki anak, dan anak tersebut mewarisi beberapa perilaku seperti orang tuanya, pada warna rambut, warna kulit, wajah, dll.

Latihan Mandiri

1. Buat sebuah Parent Class bernama Kendaraan di file latihan_4.py.

   a. Class ini harus memiliki constructor dengan atribut privat: _ _merk, _ _tahun_produksi, dan _ _warna.
   b. Miliki method tampilkan_info() yang mencetak semua informasi dasar kendaraan tersebut.
   c. Miliki method nyalakan_mesin() yang mencetak pesan umum seperti "Mesin kendaraan menyala."

```python
class Kendaraan:
    def __init__(self,merk,tahun_produksi,warna):
        self.__merk = merk
        self.__tahun_produksi = tahun_produksi
        self.__warna = warna

    def tampilkan_info(self):
        print(f"\nMerk: {self.__merk}, Tahun Produksi: {self.__tahun_produksi}, Warna: {self.__warna}")

    def nyalakan_mesin(self):
        print("\nMesin kendaraan dinyalakan.")


```

2. Buat dua Child Classes yang mewarisi dari Kendaraan: Mobil dan Motor.

3. Untuk Class Mobil

   a. Tambahkan atribut privat baru di constructor-nya: _ _jumlah_pintu.
   b. Gunakan super()._ _init_ _() untuk menginisialisasi atribut warisan.
   c. Lakukan override pada method tampilkan_info() untuk menyertakan informasi jumlah_pintu.
   d. Buat method baru yang spesifik untuk mobil, misalnya buka_pintu_bagasi().

EMAIL : itsmalik57@gmail.com

```python
class Mobil(Kendaraan):
    def __init__(self, merk, tahun_produksi, warna, jumlah_pintu):
        super().__init__(merk, tahun_produksi, warna)
        self.__jumlah_pintu = jumlah_pintu

    def tampilkan_info(self):
        super().tampilkan_info()
        print(f"\nJumlah Pintu: {self.__jumlah_pintu}")

    def buka_pintu_bagasi(self):
        print("\nPintu bagasi mobil dibuka.")
```

4. Untuk Class Motor

a. Tambahkan atribut privat baru di constructor-nya: _ _kapasitas_tangki.
b. Gunakan super()._ _init_ _() untuk menginisialisasi atribut warisan.
c. Lakukan override pada method nyalakan_mesin() untuk mencetak pesan yang lebih spesifik, misalnya "Brmm... Mesin motor dinyalakan dengan kick starter!".

```python
class Motor(Kendaraan):
    def __init__(self, merk, tahun_produksi, warna, kapasitas_tangki):
        super().__init__(merk, tahun_produksi, warna)
        self.__kapasitas_tangki = kapasitas_tangki

    def tampilkan_info(self):
        super().tampilkan_info()
        print(f"\nKapasitas Tangki: {self.__kapasitas_tangki} liter")

    def nyalakan_mesin(self):
        super().nyalakan_mesin()
        print("\nBrmm... Mesin motor dinyalakan dengan kick starter!")


```

5. Di bagian utama program Anda

a. Buat satu objek Mobil (misalnya, Toyota Avanza).
b. Buat satu objek Motor (misalnya, Honda Beat).
c. Panggil semua method dari kedua objek tersebut (tampilkan_info(), nyalakan_mesin(), dan method spesifik lainnya) untuk menunjukkan bahwa inheritance dan overriding berjalan dengan benar.

```python
mobil1 = Mobil("Toyota", 2020, "Merah", 4)
motor1 = Motor("Yamaha", 2019, "Hitam", 15)

mobil1.tampilkan_info()
mobil1.nyalakan_mesin()
mobil1.buka_pintu_bagasi()

motor1.tampilkan_info()
motor1.nyalakan_mesin()
```