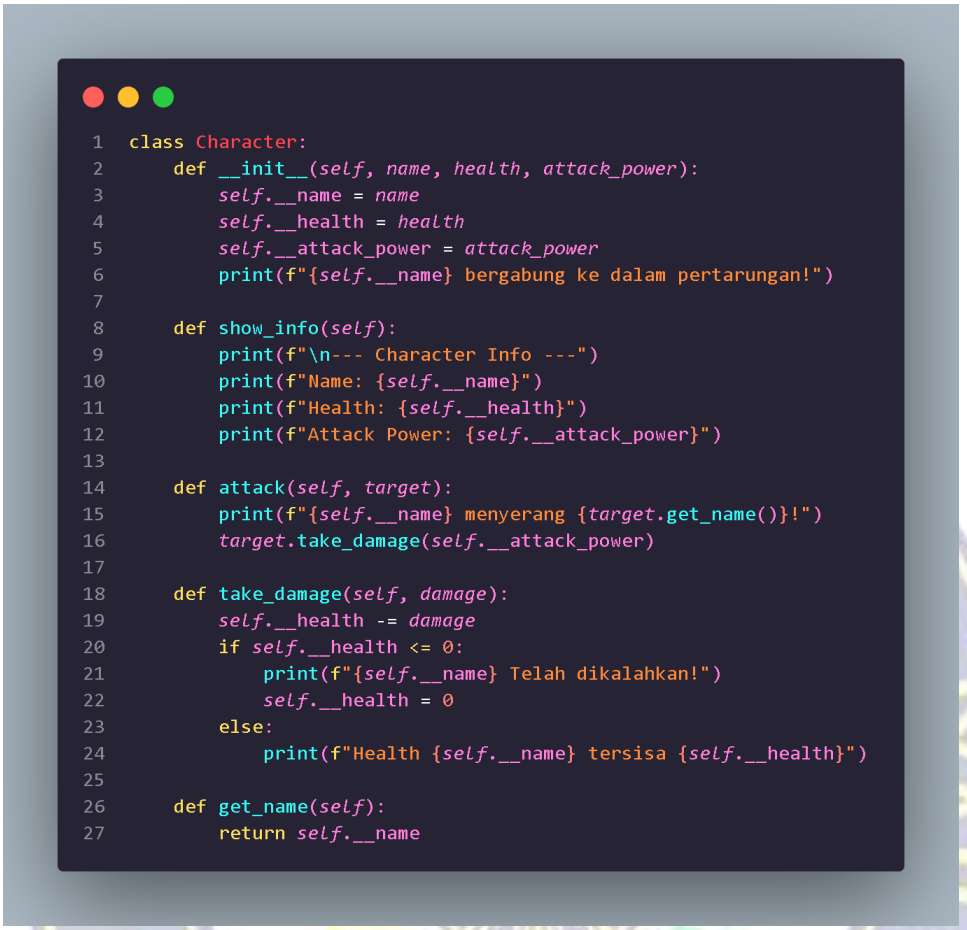
LAPORAN PRAKTIKUM PEMOGRAMAN BERORIERNTASI OBJEK (PBO)
PRAKTIKUM 4

2411102441249
HERVINO ISLAMI FASHA

FAKULTAS SAINS DAN TEKNOLOGI
PROGRAM STUDI S1 TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

2411102441249@umkt.ac.id

1. Membuat Parent Class Character



```python
class Character:
    def __init__(self, name, health, attack_power):
        self.__name = name
        self.__health = health
        self.__attack_power = attack_power
        print(f"{self.__name} bergabung ke dalam pertarungan!")

    def show_info(self):
        print(f"\n--- Character Info ---")
        print(f"Name: {self.__name}")
        print(f"Health: {self.__health}")
        print(f"Attack Power: {self.__attack_power}")

    def attack(self, target):
        print(f"{self.__name} menyerang {target.get_name()}!")
        target.take_damage(self.__attack_power)

    def take_damage(self, damage):
        self.__health -= damage
        if self.__health <= 0:
            print(f"{self.__name} Telah dikalahkan!")
            self.__health = 0
        else:
            print(f"Health {self.__name} tersisa {self.__health}")

    def get_name(self):
        return self.__name
```

A. Class dan Constructor

class Character:

- __init__ adalah constructor, dipanggil saat objek baru dibuat.
- self.__name, self.__health, self.__attack_power adalah atribut privat (pakai __).
- Saat karakter dibuat, muncul pesan “[nama] bergabung ke dalam pertarungan!”.

B. Menampilkan info karakter

def show_info(self):

print(f"\n--- Character Info ---")

print(f'Name: {self.__name}")

print(f'Health: {self.__health}")

print(f'Attack Power: {self.__attack_power}")

- Method untuk menampilkan informasi karakter: nama, health, dan attack power.

C. Menyerang karakter lain

```
def attack(self, target):
    print(f"{self.__name} menyerang {target.get_name()}!")
    target.take_damage(self.__attack_power)
```

- Method attack dipakai untuk menyerang target.
- Target dipaksa menerima damage sebesar self.__attack_power. target.get_name() dipanggil untuk ambil nama lawan.

D. Menerima damage

```
def take_damage(self, damage):
    self.__health -= damage
    if self.__health <= 0:
        print(f"{self.__name} Telah dikalahkan!")
        self.__health = 0
    else:
        print(f"Health {self.__name} tersisa {self.__health}")
```

- Method take_damage mengurangi health.
- Jika health jatuh sampai 0 atau kurang, karakter dianggap kalah.
- Jika masih ada sisa health, ditampilkan berapa yang tersisa.

E. Mengambil Nama Karakter

```
def get_name(self):
    return self.__name
```

- Memberikan akses ke atribut __name (karena bersifat privat).

Output:

```
Aragorn menyerang Gimli!
Health Gimli tersisa 110
Gimli menyerang Aragorn!
Health Aragorn tersisa 80
```

F. Membuat Child Class Warrior

aragorn = Character("Aragorn", 100, 15)

aragorn.show_info()

- Membuat objek aragorn dari class Character.

  Nama: "Aragorn"

  Health: 100

  Attack Power: 15

- Constructor __init__ akan mencetak "Aragorn bergabung ke dalam pertarungan!".
- show_info() menampilkan detail karakter:

G. Membuat Objek dari Child Class

gimli = Warrior("Gimli", 120, 20, 5)

gimli.show_info()

- Membuat objek gimli dari class Warrior (child dari Character).
- Constructor Character dipanggil dulu (super()), lalu constructor Warrior.

H. Pertarungan dimulai

aragorn.attack(gimli)

gimli.attack(aragorn)

- Aragorn menyerang Gimli:

  Print: "Aragorn menyerang Gimli!"

  Gimli menerima damage 15. Karena Gimli punya armor 5, damage yang masuk = 15 - 5 = 10.

  Health Gimli berkurang jadi 110

- Gimli menyerang Aragorn:

  Print: "Gimli menyerang Aragorn!"

Aragorn menerima damage 20 (tidak ada armor).

Health Aragorn berkurang jadi 80

I. Memperluas Child Class dengan super ()

```
class Warrior(Character):
    def __init__(self, name, health, attack_power, armor):
        super().__init__(name, health, attack_power)
        self._armor = armor
        print(f"Armor set to: {self._armor}")

```

1. Deklarasi Class

class Warrior(Character):

- Warrior adalah **child class** yang mewarisi dari class Character.
- Dengan inheritance, Warrior otomatis punya atribut dan method dari Character.

2. Constructor __init__

def __init__(self, name, health, attack_power, armor):

- Constructor menerima 4 parameter:

  name → nama karakter.

  health → jumlah nyawa/HP.

  attack_power → kekuatan serangan.

  armor → tambahan atribut khusus untuk Warrior.

3. Pemanggilan constructor parent

super().__init__(name, health, attack_power)

- super() dipakai untuk memanggil constructor dari class Character.

- Jadi atribut name, health, attack_power diatur oleh class parent.

4. Penambahan atribut baru

self._armor = armor

- Warrior punya atribut tambahan bernama _armor.
- Bedanya dengan Character, hanya Warrior yang punya armor.

OUTPUT:

```
--- Membuat Objek dari Parent Class ---
Aragorn bergabung ke dalam pertarungan!

--- Character Info ---
Name: Aragorn
Health: 100
Attack Power: 15

--- Membuat Objek dari Child Class ---
Gimli bergabung ke dalam pertarungan!
Armor set to: 5

--- Character Info ---
Name: Gimli
Health: 120
Attack Power: 20
Armor: 5

--- Pertarungan Dimulai ---
Aragorn menyerang Gimli!
Health Gimli tersisa 110
Gimli menyerang Aragorn!
Health Aragorn tersisa 80
```

TUGAS

A. Code

```python
# Parent Class
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"Merk: {self.__merk}")
        print(f"Tahun Produksi: {self.__tahun_produksi}")
        print(f"Warna: {self.__warna}")

    def nyalakan_mesin(self):
        print("Mesin kendaraan menyala.")


# Child Class Mobil
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):  # override
        super().tampilkan_info()
        print(f"Jumlah Pintu: {self.__jumlah_pintu}")

    def buka_pintu_bagasi(self):
        print("Pintu bagasi mobil dibuka.")


# Child Class Motor
class Motor(Kendaraan):
    def __init__(self, merk, tahun_produksi, warna, kapasitas_tangki):
        super().__init__(merk, tahun_produksi, warna)
        self.__kapasitas_tangki = kapasitas_tangki

    def nyalakan_mesin(self):  # override
        print("Brmm... Mesin motor dinyalakan dengan kick starter!")

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


# Bagian utama program
if __name__ == "__main__":
    # Membuat objek Mobil
    mobil1 = Mobil("Mazda RX-7", 2002, "Hitam", 4)
    print("=== Info Mobil ===")
    mobil1.tampilkan_info()
    mobil1.nyalakan_mesin()
    mobil1.buka_pintu_bagasi()

    print("\n=== Info Motor ===")
    # Membuat objek Motor
    motor1 = Motor("Honda Beat", 2022, "Merah", 4)
    motor1.tampilkan_info()
    motor1.nyalakan_mesin()

```

B. Output

```
D:\smt 3\PBO\P4>C:/Python313/python.exe "d:/smt 3/PBO/P4/tugas_4.py"
=== Info Mobil ===
Merk: Mazda RX-7
Tahun Produksi: 2002
Warna: Hitam
Jumlah Pintu: 4
Mesin kendaraan menyala.
Pintu bagasi mobil dibuka.

=== Info Motor ===
Merk: Honda Beat
Tahun Produksi: 2022
Warna: Merah
Kapasitas Tangki: 4 liter
Brmm... Mesin motor dinyalakan dengan kick starter!
```

C. Refleksi

Contoh lain dari kehidupan sehari-hari adalah sistem alat elektronik. Kelas induk dapat berupa Elektronik yang memiliki atribut seperti merek, daya, dan garansi. Kelas anak seperti Televisi dan Kulkas bisa mewarisi atribut ini, kemudian menambahkan fitur khusus seperti ukuran layar untuk televisi atau kapasitas penyimpanan untuk kulkas.