

# **LAPORAN PRAKTIKUM**

**PEMROGRAMAN BERORIENTASI OBJEK (PBO) – [TUGAS 2]**



**Disusun Oleh**  
Reza Chairul Manam  
120140086

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNOLOGI INDUSTRI  
INSTITUT TEKNOLOGI SUMATERA  
2025**

# 1 Soal Nomor 1

## 1.1 Input Soal

Buat sebuah program yang menerapkan konsep pewarisan (inheritance) dan enkapsulasi (encapsulation) dengan studi kasus berikut:

### 1.1.1 Deskripsi Umum

Program ini akan mensimulasikan hirarki kendaraan dengan tiga kelas:

- Kelas induk **Kendaraan (A)**
- Kelas turunan **Mobil (B)**, yang merupakan turunan dari Kendaraan
- Kelas turunan **MobilSport (C)**, yang merupakan turunan dari Mobil

Pada kelas `MobilSport`, gunakan konsep enkapsulasi dengan menerapkan getter dan setter untuk propertinya.

### 1.1.2 1. Kelas Kendaraan (Kelas A - Induk)

#### Properti:

- `jenis`: menyimpan jenis kendaraan, misalnya "Darat", "Air", atau "Udara".
- `kecepatan_maksimum`: menyimpan kecepatan maksimum dalam km/-jam.

#### Method:

- `info_kendaraan()`: Menampilkan informasi kendaraan.
- `bergerak()`: Menampilkan pesan bahwa kendaraan sedang bergerak.

### 1.1.3 2. Kelas Mobil (Kelas B - Turunan dari Kendaraan)

#### Properti:

- `merk`: menyimpan merek mobil, misalnya "Toyota", "BMW", dll.
- `jumlah_pintu`: menyimpan jumlah pintu mobil.

#### Method:

- `info_mobil()`: Menampilkan informasi tentang mobil.
- `bunyikan_klakson()`: Menampilkan suara klakson mobil.

### 1.1.4 3. Kelas MobilSport (Kelas C - Turunan dari Mobil)

#### Properti (Menggunakan Enkapsulasi - Private):

- `__tenaga_kuda`: menyimpan jumlah tenaga kuda mobil sport.
- `__harga`: menyimpan harga mobil sport dalam juta rupiah.

#### Method:

- `get_tenaga_kuda()`: Getter untuk mengambil nilai `__tenaga_kuda`.
- `set_tenaga_kuda(value)`: Setter untuk mengatur nilai `__tenaga_kuda`.
- `get_harga()`: Getter untuk mengambil nilai `__harga`.
- `set_harga(value)`: Setter untuk mengatur nilai `__harga`.
- `info_mobil_sport()`: Menampilkan informasi tentang mobil sport.
- `mode_balap()`: Menampilkan pesan bahwa mobil sport masuk ke mode balap.

## 1.2 Source Code

```
# Latihan:
class Kendaraan:
    def __init__(self, jenis, kecepatan_maksimum):
        self.jenis = jenis
        self.kecepatan_maksimum = kecepatan_maksimum

    def info_kendaraan(self):
        print(f"Jenis Kendaraan: {self.jenis}")
        print(f"Kecepatan Maksimum: {self.kecepatan_maksimum} km/jam")

    def bergerak(self):
        print("Kendaraan sedang bergerak...")

class Mobil(Kendaraan):
    def __init__(self, jenis, kecepatan_maksimum, merk, jumlah_pintu):
        super().__init__(jenis, kecepatan_maksimum)
        self.merk = merk
        self.jumlah_pintu = jumlah_pintu
```

```

def info_mobil(self):
    self.info_kendaraan()
    print(f"Merek Mobil: {self.merk}")
    print(f"Jumlah Pintu: {self.jumlah_pintu}")

def bunyikan_klakson(self):
    print("Beep beep! Mobil membunyikan klakson!")

class MobilSport(Mobil):
    def __init__(self, jenis, kecepatan_maksimum, merk,
        jumlah_pintu, tenaga_kuda, harga):
        super().__init__(jenis, kecepatan_maksimum, merk,
            jumlah_pintu)
        self.__tenaga_kuda = tenaga_kuda # Private
            Attribute
        self.__harga = harga # Private Attribute

    def get_tenaga_kuda(self):
        return self.__tenaga_kuda

    def set_tenaga_kuda(self, value):
        if value > 0:
            self.__tenaga_kuda = value
        else:
            print("Tenaga kuda harus lebih dari 0!")

    def get_harga(self):
        return self.__harga

    def set_harga(self, value):
        if value > 0:
            self.__harga = value
        else:
            print("Harga harus lebih dari 0!")

    def info_mobil_sport(self):
        self.info_mobil()
        print(f"Tenaga Kuda: {self.__tenaga_kuda} HP")
        print(f"Harga: Rp {self.__harga} juta")

    def mode_balap(self):
        print("Mobil Sport masuk ke mode balap!")

```

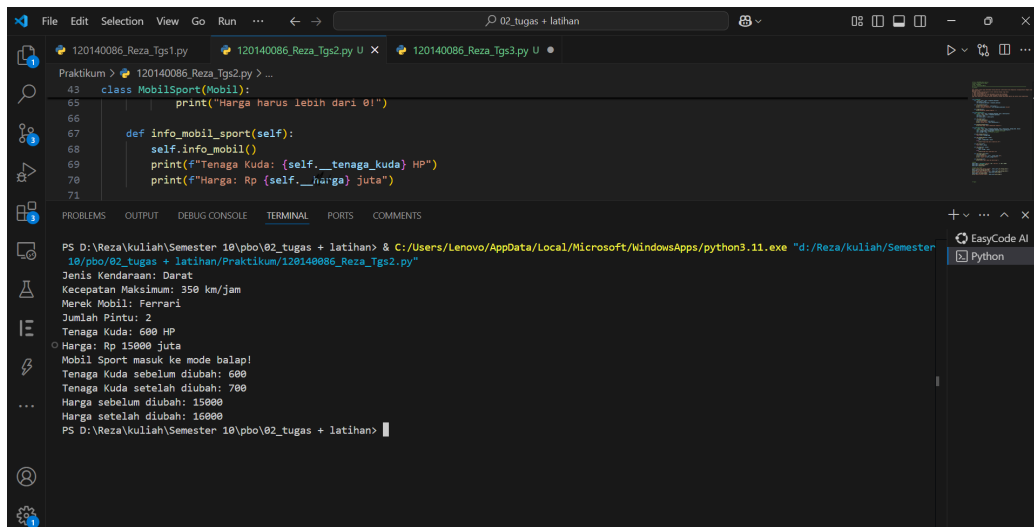
```

# Objek
mobil_sport = MobilSport("Darat", 350, "Ferrari", 2, 600,
    15000)
mobil_sport.info_mobil_sport()
mobil_sport.mode_balap()

# Getter dan setter
print("Tenaga Kuda sebelum diubah:", mobil_sport.
    get_tenaga_kuda())
mobil_sport.set_tenaga_kuda(700)
print("Tenaga Kuda setelah diubah:", mobil_sport.
    get_tenaga_kuda())
print("Harga sebelum diubah:", mobil_sport.get_harga())
mobil_sport.set_harga(16000)
print("Harga setelah diubah:", mobil_sport.get_harga())

```

### 1.3 Output Hasil



```

Praktikum > 120140086_Reza_Tgs2.py > ...
43 class MobilSport(Mobil):
65     print("Harga harus lebih dari 0!")
66
67     def info_mobil_sport(self):
68         self.info_mobil()
69         print(f"Tenaga Kuda: {self.__tenaga_kuda} HP")
70         print(f"Harga: Rp {self.__harga} juta")
71
PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan> & C:/Users/Lenovo/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/Reza/kuliah/Semester
10/pbo/02_tugas + latihan/Praktikum/120140086_Reza_Tgs2.py"
Jenis Kendaraan: Darat
Kecepatan Maksimum: 350 km/jam
Merek Mobil: Ferrari
Jumlah Pintu: 2
Tenaga Kuda: 600 HP
Harga: Rp 15000 juta
Mobil Sport masuk ke mode balap!
Tenaga Kuda sebelum diubah: 600
Tenaga Kuda setelah diubah: 700
Harga sebelum diubah: 15000
Harga setelah diubah: 16000
PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan>

```

### 1.4 Penjelasan

Program ini menerapkan konsep pewarisan (*inheritance*) dan enkapsulasi (*encapsulation*) dalam simulasi hirarki kendaraan. Terdapat tiga kelas utama: Kendaraan sebagai kelas induk, Mobil sebagai turunan dari Kendaraan, dan MobilSport sebagai turunan dari Mobil. Kelas Kendaraan memiliki atribut jenis dan kecepatan\_maksimum serta metode info\_kendaraan() dan bergerak().

Kelas Mobil menambahkan atribut merk dan jumlah\_pintu, serta metode tambahan `info_mobil()` dan `bunyikan_klakson()`. Pada kelas `MobilSport`, konsep enkapsulasi diterapkan dengan atribut privat `__tenaga_kuda` dan `__harga`, yang hanya dapat diakses melalui metode getter dan setter (`get_tenaga_kuda()`, `set_tenaga_kuda()`, `get_harga()`, dan `set_harga()`). Program ini juga menunjukkan cara penggunaan getter dan setter untuk mengubah serta mengambil nilai atribut yang dienkapsulasi. Objek `MobilSport` kemudian dibuat dan menjalankan berbagai metode untuk menampilkan informasi serta mengaktifkan mode balap.

## 2 Soal Nomor 2

### 2.1 Input Soal

Minggu ini hanya terdiri dari 1 Problem Set. Kalian perlu membuat sebuah permainan sederhana tentang pertarungan Robot.

Kalian akan membuat kelas `Robot` yang terdiri dari beberapa properti seperti `attack`, `Hp`, dll., serta beberapa metode seperti `attack_enemy()` atau `regen_health()`. Permainan ini akan berakhir ketika salah satu robot memiliki `Hp = 0`.

Kalian bisa lebih kreatif dengan menambahkan konsep seperti `attack_accuracy` agar serangan dapat meleset dalam beberapa kesempatan atau menambahkan mekanisme skill seperti `stun`, `silence`, dll., pada musuh. (Bagian ini opsional).

#### 2.1.1 Struktur Kelas

- **Kelas Robot:** Berisi mekanisme `attack`, `hp`, dan mekanisme pertarungan robot.
- **Kelas Game:** Berfungsi untuk menentukan jumlah ronde serta mengatur jalannya permainan.

#### 2.1.2 Contoh Output Permainan

```
Round-1
=====

Atreus [500|10]
Daedalus [750|8]

1. Attack      2. Defense      3. Giveup
Atreus, pilih aksi: 1

1. Attack      2. Defense      3. Giveup
Daedalus, pilih aksi: 1

----- Daedalus gagal menyerang -----

Round-2
=====

Atreus [500|10]
Daedalus [666|7]
```

```
1. Attack      2. Defense      3. Giveup
Atreus, pilih aksi: 3

1. Attack      2. Defense      3. Giveup
Daedalus, pilih aksi: 1

Daedalus menang!
```

## 2.2 Source Code

```
# Tugas
import random

class Robot: # Kelas Robot
    def __init__(self, name, hp, attack_power):
        self.name = name
        self.hp = hp
        self.attack_power = attack_power

    def attack_enemy(self, enemy): # Metode untuk
        menghancurkan musuh
        if random.random() > 0.2: # 80% chance to hit
            damage = random.randint(self.attack_power - 2,
                                    self.attack_power + 2)
            enemy.hp -= damage
            print(f"{self.name} menyerang {enemy.name} dan
                  memberikan {damage} damage!")
        else:
            print(f"{self.name} gagal menyerang {enemy.name}
                  {!}")

    def regen_health(self): # Metode untuk meregenerasi HP
        heal = random.randint(5, 15)
        self.hp += heal
        print(f"{self.name} meregenerasi {heal} HP!")

    def is_alive(self): # Metode untuk mengecek apakah
        robot masih hidup
        return self.hp > 0

class Game: # Kelas Game
    def __init__(self, robot1, robot2):
```



```

self.robot1 = robot1
self.robot2 = robot2
self.round = 1

def start(self): # Metode untuk memulai pertarungan
    while self.robot1.is_alive() and self.robot2.is_alive(): # Looping selama kedua robot masih hidup
        print(f"\nRound-{self.round}
        =====")
        print(f"{self.robot1.name} [{self.robot1.hp}|{self.robot1.attack_power}]")
        print(f"{self.robot2.name} [{self.robot2.hp}|{self.robot2.attack_power}]")

        for robot, enemy in [(self.robot1, self.robot2), (self.robot2, self.robot1)]: # Bergantian menyerang
            if not robot.is_alive() or not enemy.is_alive(): # Jika salah satu robot sudah mati, break
                break
            action = input(f"{robot.name}, pilih aksi: 1. Attack 2. Regen 3. Giveup\n") # Pilihan aksi
            if action == "1":
                robot.attack_enemy(enemy)
            elif action == "2":
                robot.regen_health()
            elif action == "3":
                print(f"{robot.name} menyerah! {enemy.name} menang!")
                return
            self.round += 1

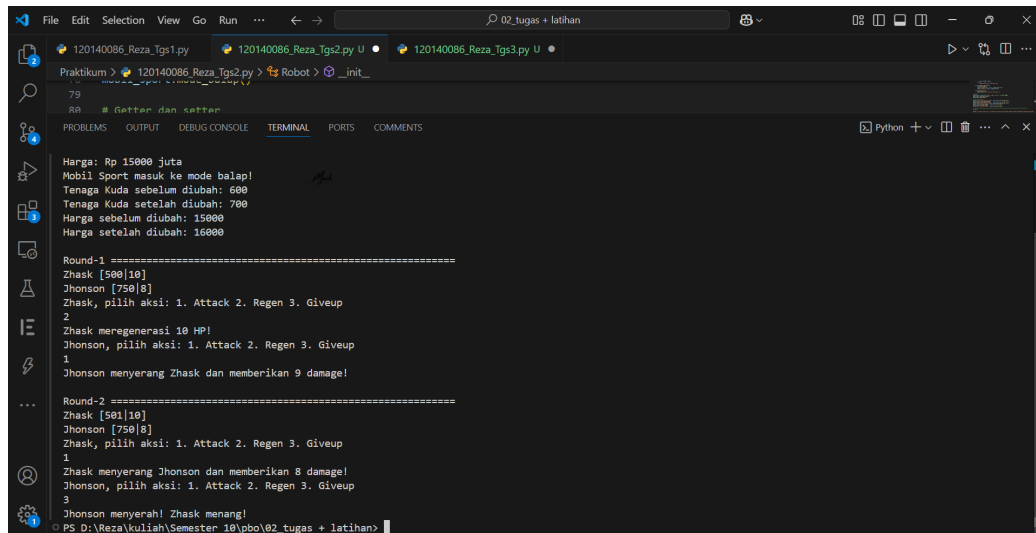
    winner = self.robot1 if self.robot1.is_alive() else self.robot2 # Menentukan pemenang
    print(f"\n{winner.name} menang!")

# Objek Robot
robot1 = Robot("Zhask", 500, 10)
robot2 = Robot("Jhonson", 750, 8)

```

```
battle = Game(robot1, robot2) # Objek Game
battle.start() # Memulai pertarungan
```

## 2.3 Output Hasil



```
Praktikum > 120140086_Reza_Tgs1.py 120140086_Reza_Tgs2.py U 120140086_Reza_Tgs3.py U
79
80 # Getter dan setter
81
82 Harga: Rp 15000 juta
83 Mobil Sport masuk ke mode balap!
84 Tenaga Kuda sebelum diubah: 600
85 Tenaga Kuda setelah diubah: 700
86 Harga sebelum diubah: 15000
87 Harga setelah diubah: 16000
88
89 Round-1 =====
90 Zhask [500|10]
91 Jhonson [750|8]
92 Zhask, pilih aksi: 1. Attack 2. Regen 3. Giveup
93 2
94 Zhask meregenerasi 10 HP!
95 Jhonson, pilih aksi: 1. Attack 2. Regen 3. Giveup
96 1
97 Jhonson menyerang Zhask dan memberikan 9 damage!
98
99 Round-2 =====
100 Zhask [501|10]
101 Jhonson [750|8]
102 Zhask, pilih aksi: 1. Attack 2. Regen 3. Giveup
103 1
104 Zhask menyerang Jhonson dan memberikan 8 damage!
105 Jhonson, pilih aksi: 1. Attack 2. Regen 3. Giveup
106 3
107 Jhonson menyerah! Zhask menang!
108
109 PS D:\Reza\kuliah\Semester 10\pbo\02_tugas + latihan>
```

## 2.4 Penjelasan

Program ini merupakan permainan sederhana tentang pertarungan robot yang menerapkan konsep OOP (*Object-Oriented Programming*). Terdapat dua kelas utama: Robot dan Game. Kelas Robot memiliki atribut seperti name, hp (jumlah nyawa), dan attack\_power (kekuatan serangan). Selain itu, kelas ini juga memiliki metode attack\_enemy() untuk menyerang lawan dengan peluang keberhasilan 80%, regen\_health() untuk memulihkan nyawa secara acak, serta is\_alive() untuk mengecek apakah robot masih hidup. Kelas Game digunakan untuk mengatur jalannya permainan. Dalam metode start(), pertarungan berlangsung dalam bentuk ronde, di mana masing-masing robot diberi kesempatan untuk menyerang atau memulihkan nyawa. Pemain dapat memilih aksi melalui input, yaitu menyerang, melakukan regenerasi, atau menyerah. Permainan berakhir ketika salah satu robot memiliki hp bernilai nol atau ada robot yang menyerah. Pemenang akan diumumkan berdasarkan robot yang masih hidup. Program ini mengimplementasikan elemen interaktif dengan input pengguna, serta elemen acak dalam serangan dan regenerasi untuk menambah variasi dalam pertarungan.

### 3 Lampiran

Pada bagian ini disertakan tautan yang berkaitan dengan tugas praktikum. Tautan GitHub berisi kode sumber yang telah dibuat selama praktikum, sedangkan tautan Overleaf merupakan tempat penyusunan laporan ini.

**Repository GitHub Tugas:** Berikut adalah tautan menuju repository GitHub yang berisi seluruh kode sumber tugas praktikum:

<https://github.com/rezachairul/Latihan-PBO-2025/tree/main/Praktikum>

**Dokumen Laporan di Overleaf:** Untuk melihat laporan ini secara langsung di Overleaf, gunakan tautan berikut:

<https://www.overleaf.com/read/jyxrzkdcnzqx#0ffb83>