PRAKTIKUM ENCAPSULATION & INHERITANCE

MINGGU KE – 7

Mata Kuliah

Pemrograman Berbasis Objek

Oleh:

tafrih humaidi

24091397030

2024B

PROGRAM STUDI D4 MANAJEMEN INFORMATIKA

FAKULTAS VOKASI

UNIVERSITAS NEGERI SURABAYA

2025

1. **Latihan Encapsulasi (Inventaris Laptop)**

INPUT CODE PROGRAM

```python
class Laptop:
    def __init__(self, merek, model, tahun, spesifikasi, harga_beli):
        self.merek = merek
        self.model = model
        self.tahun = tahun
        self.spesifikasi = spesifikasi
        self._harga_beli = harga_beli  # atribut private

    # Getter -> untuk membaca harga beli
    def get_harga_beli(self):
        return self._harga_beli

    # Setter -> untuk mengubah harga beli dengan validasi
    def set_harga_beli(self, harga_baru):
        if harga_baru > 0:
            self._harga_beli = harga_baru
        else:
            print("Harga beli harus lebih besar dari 0!!")

    # Hitung harga jual (harga beli + 25% dari harga beli)
    def hitung_harga_jual(self):
        return self._harga_beli + (0.25 * self._harga_beli)

    # Tampilkan informasi laptop
    def tampil_info(self):
        print(f"Laptop: {self.merek} {self.model}")
        print(f"Tahun: {self.tahun}")
        print(f"Spesifikasi: {self.spesifikasi}")
        print(f"Harga Beli: Rp {self.get_harga_beli():,.0f}")
        print(f"Harga Jual: Rp {self.hitung_harga_jual():,.0f}")
        print("-" * 40)
```

```python
        # Membuat beberapa objek laptop
laptop1 = Laptop("Asus", "VivoBook 14", 2022, "i5, 8GB RAM, 512GB SSD", 8000000)
laptop2 = Laptop("Lenovo", "ThinkPad X1", 2021, "i7, 16GB RAM, 1TB SSD", 15000000)
laptop3 = Laptop("HP", "Pavilion 15", 2020, "Ryzen 5, 8GB RAM, 512GB SSD", 10000000)

# Tampilkan informasi setiap laptop
laptop1.tampil_info()
laptop2.tampil_info()
laptop3.tampil_info()

# Hitung total harga beli & jual semua laptop
total_beli = (
    laptop1.get_harga_beli() +
    laptop2.get_harga_beli() +
    laptop3.get_harga_beli()
)

total_jual = (
    laptop1.hitung_harga_jual() +
    laptop2.hitung_harga_jual() +
    laptop3.hitung_harga_jual()
)

print(f"Total Harga Beli: Rp {total_beli:,.0f}")
print(f"Total Harga Jual: Rp {total_jual:,.0f}")

# Ubah harga beli laptop melalui method setter
print("\n== Ubah harga beli laptop1 ==")
laptop1.set_harga_beli(8500000) # ubah harga beli
laptop1.tampil_info()

# Coba setter dengan nilai salah
laptop1.set_harga_beli(-5000000) # seharusnya muncul pesan error
```

OUTPUT CODE PROGRAM

```
Laptop: Asus VivoBook 14
Tahun: 2022
Spesifikasi: i5, 8GB RAM, 512GB SSD
Harga Beli: Rp 8,000,000
Harga Jual: Rp 10,000,000
----------------------------------------
Laptop: Lenovo ThinkPad X1
Tahun: 2021
Spesifikasi: i7, 16GB RAM, 1TB SSD
Harga Beli: Rp 15,000,000
Harga Jual: Rp 18,750,000
----------------------------------------
Laptop: HP Pavilion 15
Tahun: 2020
Spesifikasi: Ryzen 5, 8GB RAM, 512GB SSD
Harga Beli: Rp 10,000,000
Harga Jual: Rp 12,500,000
----------------------------------------
Total Harga Beli: Rp 33,000,000
Total Harga Jual: Rp 41,250,000

== Ubah harga beli laptop1 ==
Laptop: Asus VivoBook 14
Tahun: 2022
Spesifikasi: i5, 8GB RAM, 512GB SSD
Harga Beli: Rp 8,500,000
Harga Jual: Rp 10,625,000
----------------------------------------
Harga beli harus lebih besar dari 0!!
```

Penjelasan source code

```python
class Laptop:
    def __init__(self, merek, model, tahun, spesifikasi, harga_beli):
        self.merek = merek
        self.model = model
        self.tahun = tahun
        self.spesifikasi = spesifikasi
        self._harga_beli = harga_beli  # atribut private
```

Kode ini memiliki class yang berisi beberapa atribut seperti merek, tahun, spesifikasi, dan harga beli.

Kode harga_beli menggunakan ._____karena bersifat private yang hanya bisa diakses melalui method di dalam calss, bukan dari luar objek. Hal ini dibuat untuk menjaga keamanan data harga dari perubahan langsung oleh pengguna.

```python
# Getter -> untuk membaca harga beli
def get_harga_beli(self):
    return self._harga_beli

# Setter -> untuk mengubah harga beli dengan validasi
def set_harga_beli(self, harga_baru):
    if harga_baru > 0:
        self._harga_beli = harga_baru
    else:
        print("Harga beli harus lebih besar dari 0!!")

# Hitung harga jual (harga beli + 25% dari harga beli)
def hitung_harga_jual(self):
    return self._harga_beli + (0.25 * self._harga_beli)
```

Kode Python ini menunjukkan implementasi enkapsulasi dalam sebuah kelas melalui tiga metode (method) fungsionalitas harga: get_harga_beli() berfungsi sebagai Getter yang aman untuk membaca nilai atribut *private* _harga_beli; set_harga_beli() bertindak sebagai Setter yang memungkinkan perubahan nilai harga beli hanya jika nilai harga_baru lebih besar dari nol, menegakkan validasi data; dan hitung_harga_jual() menghitung harga jual dengan menambahkan margin keuntungan sebesar 25% dari harga beli, menyembunyikan logika perhitungan dari luar kelas.

```python
# Tampilkan informasi laptop
def tampil_info(self):
    print(f"Laptop: {self.merek} {self.model}")
    print(f"Tahun: {self.tahun}")
    print(f"Spesifikasi: {self.spesifikasi}")
    print(f"Harga Beli: Rp {self.get_harga_beli():,.0f}")
    print(f"Harga Jual: Rp {self.hitung_harga_jual():,.0f}")
    print("-" * 40)
```

Kode tersebut mendefinisikan metode tampil_info() yang berfungsi untuk menampilkan semua detail informasi tentang objek laptop (merek, model, tahun, dan spesifikasi). Metode ini secara khusus memanggil get_harga_beli() dan hitung_harga_jual() untuk mengambil dan menampilkan harga beli serta harga jual laptop.

```python
# Tampilkan informasi setiap laptop
laptop1.tampil_info()
laptop2.tampil_info()
laptop3.tampil_info()

# Hitung total harga beli & jual semua laptop
total_beli = (
    laptop1.get_harga_beli() +
    laptop2.get_harga_beli() +
    laptop3.get_harga_beli()
)

total_jual = (
    laptop1.hitung_harga_jual() +
    laptop2.hitung_harga_jual() +
    laptop3.hitung_harga_jual()
)

print(f"Total Harga Beli: Rp {total_beli:,.0f}")
print(f"Total Harga Jual: Rp {total_jual:,.0f}")
```

Kode Python ini melakukan dua fungsi utama setelah objek-objek laptop (laptop1, laptop2, laptop3) dibuat. Pertama, kode memanggil metode tampil_info() untuk setiap objek, yang akan menampilkan detail dan harga jual dari masing-masing laptop. Kedua, kode ini menghitung total harga beli dengan memanggil get_harga_beli() dari setiap objek, dan menghitung total harga jual dengan memanggil hitung_harga_jual() dari setiap objek, lalu menjumlahkannya. Hasil dari kedua total perhitungan tersebut kemudian dicetak ke konsol dengan format mata uang rupiah

```
# Ubah harga beli laptop melalui method setter
print("\n== Ubah harga beli laptop1 ==")
laptop1.set_harga_beli(8500000) # ubah harga beli
laptop1.tampil_info()

# Coba setter dengan nilai salah
laptop1.set_harga_beli(-5000000) # seharusnya muncul pesan error
```

Kode di baris terakhir dibunakan untuk mencari nilai negatif dan akan muncul pesan error di terminal Kode pertama digunakan untuk mengubah harga beli laptop1 menjadi harga beli.

Kode di baris terakhir dibunakan untuk mencari nilai negatif dan akan muncul pesan error di terminal

2. Latihan Inheritance (Kendaraan)

```python
# Superclass
class Kendaraan:
    def __init__(self, merk, tahun):
        self.merk = merk
        self.tahun = tahun

    def info(self):
        print(f"Merk: {self.merk}, Tahun: {self.tahun}")

    def bergerak(self):
        print("Kendaraan ini bisa bergerak")

# Subclass Mobil
class Mobil(Kendaraan):
    def __init__(self, merk, tahun, jumlah_pintu):
        super().__init__(merk, tahun)
        self.jumlah_pintu = jumlah_pintu

    def info(self):
        super().info()
        print(f"Jumlah Pintu: {self.jumlah_pintu}")

    def bergerak(self):  # override method
        print("Mobil berjalan di jalan raya")

# Subclass Motor
class Motor(Kendaraan):
    def __init__(self, merk, tahun, jenis):
        super().__init__(merk, tahun)
        self.jenis = jenis
```

```
    def info(self):
        super().info()
        print(f"Jenis Motor: {self.jenis}")

    def bergerak(self):  # override method
        print("Motor melaju dengan cepat")

# Membuat objek
mobil1 = Mobil("Toyota", 2020, 4)
motor1 = Motor("Honda", 2021, "Matic")

# Panggil Method info() dan bergerak()
mobil1.info()
mobil1.bergerak()

print("-" * 30)

motor1.info()
motor1.bergerak()
```

OUPUT CODE PROGRAM

```
Merk: Toyota, Tahun: 2020
Jumlah Pintu: 4
Mobil berjalan di jalan raya
------------------------------
Merk: Honda, Tahun: 2021
Jenis Motor: Matic
Motor melaju dengan cepat
```

Penjelasan Code Input

```python
# Superclass
class Kendaraan:
    def __init__(self, merk, tahun):
        self.merk = merk
        self.tahun = tahun

    def info(self):
        print(f"Merk: {self.merk}, Tahun: {self.tahun}")

    def bergerak(self):
        print("Kendaraan ini bisa bergerak")
```

Kode ini mendefinisikan Superclass bernama Kendaraan. Kelas ini memiliki konstruktor (__init__) untuk menginisialisasi atribut dasar merk dan tahun, serta dua metode dasar: info() untuk menampilkan merek dan tahun kendaraan, dan bergerak() untuk menyatakan bahwa kendaraan bisa bergerak.

```python
# Membuat objek
mobil1 = Mobil("Toyota", 2020, 4)
motor1 = Motor("Honda", 2021, "Matic")

# Panggil Method info() dan bergerak()
mobil1.info()
mobil1.bergerak()

print("-" * 30)

motor1.info()
motor1.bergerak()
```

Kode Python ini memulai eksekusi program dengan menciptakan dua **objek** nyata dari *subclass*: mobil1 dari kelas Mobil ("Toyota", 2020, 4 pintu) dan motor1 dari kelas Motor ("Honda", 2021, "Matic"). Selanjutnya, kode mendemonstrasikan perilaku objek dengan memanggil metode **info()** dan **bergerak()** untuk masing-masing objek, di mana objek-objek tersebut secara otomatis menjalankan implementasi metode yang spesifik (di-override) yang sesuai dengan kelasnya sendiri, bukan yang ada di *superclass* Kendaraa

3. Praktikum (Novel Detektif)

INPUT CODE PROGRAM

```
# Class Novel dengan konsep Enkapsulasi
class Novel:
    def __init__(self, judul, pengarang, tahun_terbit, deskripsi, harga_beli):
        self._judul = judul
        self._pengarang = pengarang
        self._tahun_terbit = tahun_terbit
        self._deskripsi = deskripsi
        self._harga_beli = harga_beli

    # Getter (untuk membaca data private)
    def get_judul(self):
        return self._judul

    def get_pengarang(self):
        return self._pengarang

    def get_tahun_terbit(self):
        return self._tahun_terbit

    def get_deskripsi(self):
        return self._deskripsi

    def get_harga_beli(self):
        return self._harga_beli

    # Setter (untuk mengubah data private)
    def set_harga_beli(self, harga_baru):
        if harga_baru > 0:
            self._harga_beli = harga_baru
        else:
            print("Harga beli tidak boleh negatif!")
```

```python
    # Method untuk menghitung harga jual novel
    def hitung_harga_jual(self):
        # Harga jual = Harga beli - 20% * Harga beli
        return self._harga_beli - (0.2 * self._harga_beli)

    # Method untuk menampilkan informasi novel
    def tampil_info(self):
        print("Judul        : ", self._judul)
        print("Pengarang    : ", self._pengarang)
        print("Tahun Terbit : ", self._tahun_terbit)
        print("Deskripsi    : ", self._deskripsi)
        print("Harga Beli   : Rp ", f"{self.get_harga_beli():,.0f}")
        print("Harga Jual   : Rp ", f"{self.hitung_harga_jual():,.0f}")
        print("-" * 45)


# Membuat beberapa objek Novel
novel1 = Novel("Malice: Catatan Pembunuhan Sang Novelis", "Keigo Higashino", 2020, "Pembunuhan sang novelis
novel2 = Novel("Kesetiaan Mr. X", "Keigo Higashino", 2021, "Kisah penyelidikan pembunuhan misterius", 9900
novel3 = Novel("The Big Four", "Agatha Christie", 2018, "Empat tokoh berkuasa yang berambisi menguasai duni

# Menampilkan informasi setiap novel
print("=== DAFTAR NOVEL DETEKTIF MILIK CONAN ===")
novel1.tampil_info()
novel2.tampil_info()
novel3.tampil_info()

# Menghitung total harga beli dan harga jual semua novel
total_beli = novel1.get_harga_beli() + novel2.get_harga_beli() + novel3.get_harga_beli()
total_jual = novel1.hitung_harga_jual() + novel2.hitung_harga_jual() + novel3.hitung_harga_jual()
```

```python
# Contoh Setter
print("\n== Update Harga Beli Novel 1 ==")
novel1.set_harga_beli(80000) # Ubah harga beli menjadi 80000
novel1.tampil_info()

# Contoh setter dengan nilai salah
novel1.set_harga_beli(-50000) # Seharusnya muncul pesan error
```

OUTPUT CODE PROGRAM

```
=== DAFTAR NOVEL DETEKTIF MILIK CONAN ===
Judul        : Malice: Catatan Pembunuhan Sang Novelis
Pengarang    : Keigo Higashino
Tahun Terbit : 2020
Deskripsi    : Pembunuhan sang novelis terkenal
Harga Beli   : Rp 93,000
Harga Jual   : Rp 74,400
-----------------------------------------
Judul        : Kesetiaan Mr. X
Pengarang    : Keigo Higashino
Tahun Terbit : 2021
Deskripsi    : Kisah penyelidikan pembunuhan misterius
Harga Beli   : Rp 99,000
Harga Jual   : Rp 79,200
-----------------------------------------
Judul        : The Big Four
Pengarang    : Agatha Christie
Tahun Terbit : 2018
Deskripsi    : Empat tokoh berkuasa yang berambisi menguasai dunia
Harga Beli   : Rp 65,000
Harga Jual   : Rp 52,000
-----------------------------------------
Total Harga Beli Semua Novel : Rp 257,000
Total Harga Jual Jika Dijual : Rp 205,600

== Update Harga Beli Novel 1 ==
Judul        : Malice: Catatan Pembunuhan Sang Novelis
Pengarang    : Keigo Higashino
Tahun Terbit : 2020
Deskripsi    : Pembunuhan sang novelis terkenal
Harga Beli   : Rp 80,000
Harga Jual   : Rp 64,000
-----------------------------------------
Harga beli tidak boleh negatif!
```

Penjelasan

```python
# Class Novel dengan konsep Enkapsulasi
class Novel:
    def __init__(self, judul, pengarang, tahun_terbit, deskripsi, harga_beli):
        self._judul = judul
        self._pengarang = pengarang
        self._tahun_terbit = tahun_terbit
        self._deskripsi = deskripsi
        self._harga_beli = harga_beli
```

Kode yang disajikan mendefinisikan **kelas Novel** dengan konstruktor (__init__) yang menginisialisasi lima atribut novel: judul, pengarang, tahun terbit, deskripsi, dan harga beli. Semua atribut ini menggunakan konvensi penamaan dengan garis bawah di awal (_judul, _pengarang, dll.) untuk menandakan bahwa mereka adalah **atribut *private***, sebuah praktik yang bertujuan untuk menerapkan konsep **enkapsulasi** dalam pemrograman berorientasi objek (OOP). Metode ini memastikan data dapat diakses dan diubah secara terkontrol melalui *getter* dan *setter* yang akan didefinisikan kemudian, bukan diakses secara langsung.

```python
# Getter (untuk membaca data private)
def get_judul(self):
    return self._judul

def get_pengarang(self):
    return self._pengarang

def get_tahun_terbit(self):
    return self._tahun_terbit

def get_deskripsi(self):
    return self._deskripsi

def get_harga_beli(self):
    return self._harga_beli
```

Kode yang ditampilkan berisi serangkaian metode Getter yang semuanya berfungsi untuk membaca nilai dari atribut *private* (ditandai dengan garis bawah di awal, seperti _judul, _pengarang, dll.) dari objek Novel. Dengan mengembalikan nilai atribut *private* melalui metode public ini, kode tersebut secara efektif menjaga prinsip enkapsulasi dengan memberikan akses baca yang terkontrol ke data sensitif novel

```python
# Method untuk menghitung harga jual novel
def hitung_harga_jual(self):
    # Harga jual = Harga beli - 20% * Harga beli
    return self._harga_beli - (0.2 * self._harga_beli)
```

Kode tersebut berisi metode set_harga_beli() yang berfungsi sebagai Setter untuk mengubah nilai harga beli novel. Metode ini dilengkapi dengan validasi data yang memastikan harga_baru yang dimasukkan harus bernilai positif , dan akan menampilkan pesan error jika harga yang diberikan tidak valid (negatif atau nol)

```python
# Method untuk menampilkan informasi novel
def tampil_info(self):
    print("Judul        : ", self._judul)
    print("Pengarang    : ", self._pengarang)
    print("Tahun Terbit : ", self._tahun_terbit)
    print("Deskripsi    : ", self._deskripsi)
    print("Harga Beli   : Rp ", f"{self.get_harga_beli():,.0f}")
    print("Harga Jual   : Rp ", f"{self.hitung_harga_jual():,.0f}")
    print("-" * 45)
```

Semua blok kode yang Anda berikan (untuk Laptop dan Novel) bersama-sama mendemonstrasikan prinsip dasar Object-Oriented Programming (OOP), khususnya Enkapsulasi dan Inheritance. Kelas-kelas tersebut menggunakan konstruktor (__init__) untuk membuat objek dan menginisialisasi atribut, di mana data sensitif seperti harga ditandai sebagai *private* (_harga_beli) untuk melindungi integritas data. Perlindungan ini dipertahankan melalui metode Getter (get_harga_beli()) untuk membaca data, dan Setter (set_harga_beli()) yang memungkinkan modifikasi data *private* dengan menerapkan validasi, serta metode lain seperti hitung_harga_jual() dan tampil_info() yang menyembunyikan logika bisnis dan perhitungan di dalam kelas.

```python
# Setter (untuk mengubah data private)
def set_harga_beli(self, harga_baru):
```

```python
# Membuat beberapa objek Novel
novel1 = Novel("Malice: Catatan Pembunuhan Sang Novelis", "Keigo Higashino", 2020, "Pembunuhan sang novelis terkenal", 93000)
novel2 = Novel("Resolusi No. X", "Keigo Higashino", 2021, "Kisah penyelidikan pembunuhan misterius", 93000)
novel3 = Novel("The Big Four", "Agatha Christie", 2018, "Empat tokoh berkuasa yang berambisi menguasai dunia", 65000)
```

Kode program yang Anda berikan merupakan implementasi sederhana dari Pemrograman Berorientasi

Objek (OOP): Enkapsulasi (dalam studi kasus Laptop dan Novel) dan Inheritance (dalam studi kasus Kendaraan)

Metode set_harga_beli() berfungsi sebagai Setter untuk atribut *private* harga beli (_harga_beli) pada kelas Novel. Kode ini dilengkapi dengan validasi yang ketat: ia hanya akan memperbarui nilai _harga_beli jika harga_baru yang dimasukkan lebih besar dari nol. Jika nilai yang dimasukkan negatif atau nol, method akan memunculkan pesan error ("Harga beli tidak boleh negatif!"), sehingga menjamin integritas data dan menegakkan prinsip enkapsulasi.

```
# Menampilkan informasi setiap novel
print("=== DAFTAR NOVEL DETEKTIF MILIK CONAN ===")
novel1.tampil_info()
novel2.tampil_info()
novel3.tampil_info()

# Menghitung total harga beli dan harga jual semua novel
total_beli = novel1.get_harga_beli() + novel2.get_harga_beli() + novel3.get_harga_beli()
total_jual = novel1.hitung_harga_jual() + novel2.hitung_harga_jual() + novel3.hitung_harga_jual()

print(f"Total Harga Beli Semua Novel : Rp {total_beli:,.0f}")
print(f"Total Harga Jual Jika Dijual : Rp {total_jual:,.0f}")
```

Program ini mengilustrasikan dua konsep utama OOP menggunakan Python: Enkapsulasi pada studi kasus Laptop dan Novel, serta Inheritance pada studi kasus Kendaraan. Pada Enkapsulasi, data sensitif seperti _harga_beli disembunyikan menggunakan konvensi *private*, dan aksesnya diatur melalui Getter (get_harga_beli()) dan Setter (set_harga_beli()) yang mencakup validasi data untuk mencegah nilai negatif. Logika bisnis untuk menghitung harga jual juga dienkapsulasi dalam metode hitung_harga_jual(). Sementara itu, Inheritance ditunjukkan oleh Superclass Kendaraan yang diwarisi oleh Subclass Mobil dan Motor, di mana *subclass* menambahkan atribut spesifik dan menggunakan teknik Polymorphism (override) untuk menyesuaikan perilaku metode bergerak() dan info().

```
# Menghitung total harga beli dan harga jual semua novel
total_beli = novel1.get_harga_beli() + novel2.get_harga_beli() + novel3.get_harga_beli()
total_jual = novel1.hitung_harga_jual() + novel2.hitung_harga_jual() + novel3.hitung_harga_jual()

print(f"Total Harga Beli Semua Novel : Rp {total_beli:,.0f}")
print(f"Total Harga Jual Jika Dijual : Rp {total_jual:,.0f}")

# Contoh Setter
print("\n== Update Harga Beli Novel 1 ==")
novel1.set_harga_beli(80000) # Ubah harga beli menjadi 80000
novel1.tampil_info()

# Contoh setter dengan nilai salah
novel1.set_harga_beli(-50000) # Seharusnya muncul pesan error
```

Metode tampil_info() pada kelas Novel berfungsi untuk menampilkan seluruh informasi detail objek secara rapi, termasuk judul, pengarang, tahun terbit, dan deskripsi. Untuk menampilkan Harga Beli dan Harga Jual, metode ini secara spesifik memanggil self.get_harga_beli() dan self.hitung_harga_jual(). Pemanggilan melalui *getter* (get_harga_beli()) ini menegaskan implementasi enkapsulasi, memastikan data *private* (_harga_beli) diakses melalui mekanisme yang terkontrol.