LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 3

2411102441259

Malik Sabarullah Akbar

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : itsmalik57@gmail.com

Enkapsulasi, Menyembunyikan Informasi dan Melindungi Data

1. Membuat class karyawan yang memiliki constructor dan menerima id, nama, dan gaji. Dan semua atribut harus private.

```python
class Karyawan:
    def __init__(self, id, nama, gaji):
        self.__id_karyawan = id
        self.__nama_karyawan = nama
        self.__gaji_karyawan = gaji

```

2. Pembuatan getter pada id, nama, dan gaji agar pengguna luar bisa melihat isi atribut tersebut.

```python
# Getter
    def get_id(self):
        return f"Id Karyawan: {self.__id_karyawan}"

    def get_nama(self):
        return f"Nama Karyawan: {self.__nama_karyawan}"

    def get_gaji(self):
        return f"Gaji Karyawan: {self.__gaji_karyawan}"

```

EMAIL : itsmalik57@gmail.com

3. Membuat Setter untuk atribut nama dan gaji:

a. set_nama(nama_baru): Lakukan validasi sederhana, pastikan nama baru tidak kosong (bukan string kosong "").

b. set_gaji(gaji_baru): Ini adalah bagian krusial. Lakukan validasi untuk memastikan gaji_baru adalah angka positif (lebih besar dari 0). Jika data tidak valid, tampilkan pesan eror dan jangan ubah nilai gaji.

c. Catatan. ID Karyawan diasumsikan tidak bisa diubah, jadi tidak perlu membuat setter untuk id_karyawan.

```python
# Setter
    def set_nama(self, nama):
        if nama == "":
            print("Nama tidak boleh kosong")
        else:
            self.__nama_karyawan = nama
            print("Nama karyawan berhasil diubah")

    def set_gaji(self, gaji):
        if gaji <= 0:
            print("Gaji tidak boleh negatif")
        else:
            self.__gaji_karyawan = gaji
            print("Gaji karyawan berhasil diubah")

```

4. Di bagian utama program Anda:

a. Buat sebuah objek Karyawan dengan data awal yang valid.

```python
karyawan_1 = Karyawan(1, "Andi", 5000000)
```

b. Gunakan getter untuk menampilkan informasi lengkap karyawan tersebut.

```python
print(karyawan_1.get_id())    # Output: 1
print(karyawan_1.get_nama())  # Output: Andi
print(karyawan_1.get_gaji())  # Output: 5000000
```

Output:

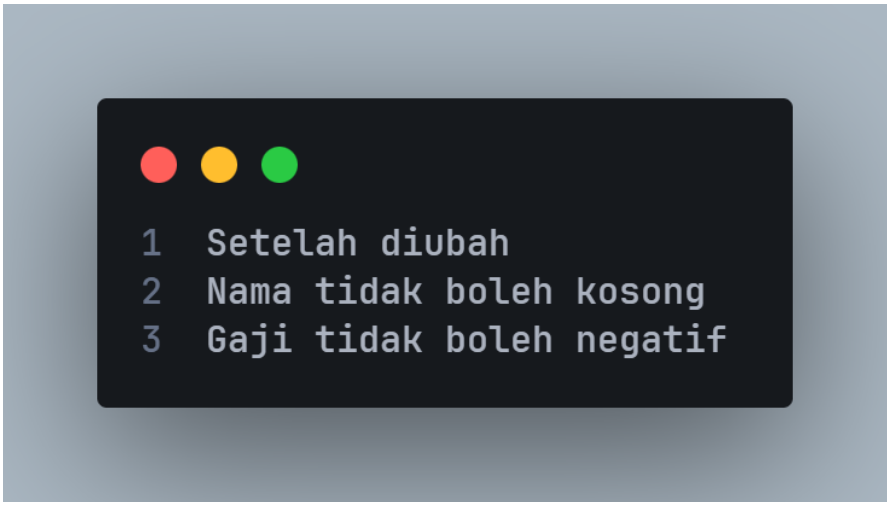
```
Id Karyawan: 1
Nama Karyawan: Andi
Gaji Karyawan: 5000000
```

c. Lakukan simulasi: Coba ubah gaji karyawan menjadi nilai negatif (misal: -5000000).
d. Amati dan pastikan program menolak perubahan ini.
e. Coba ubah nama karyawan menjadi string kosong (""). Pastikan program menolaknya.

```
print("\nSetelah diubah")

karyawan_1.set_nama("")
karyawan_1.set_gaji(-3000000)
```

Output :



```
Setelah diubah
Nama tidak boleh kosong
Gaji tidak boleh negatif
```

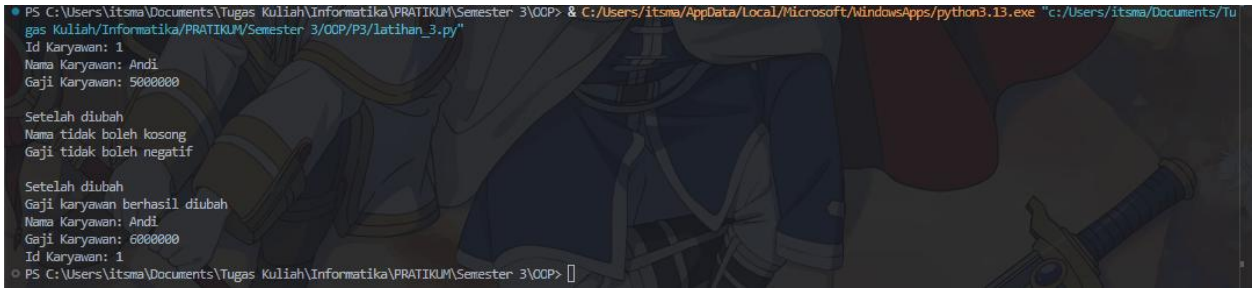
f. Lakukan perubahan gaji menjadi nilai positif yang valid (misal: 6000000).
g. Tampilkan kembali informasi lengkap karyawan untuk menunjukkan bahwa gaji
h. berhasil diperbarui dengan aman.

```python
print("\nSetelah diubah")
karyawan_1.set_gaji(6000000)
print(karyawan_1.get_nama())  # Output: Andi
print(karyawan_1.get_gaji())  # Output: 6000000
print(karyawan_1.get_id())    # Output: 1
```

Output :

```
Setelah diubah
Gaji karyawan berhasil diubah
Nama Karyawan: Andi
Gaji Karyawan: 6000000
Id Karyawan: 1
```

Final Output :



EMAIL : itsmalik57@gmail.com

Kesimpulan

Encapsulation adalah pilar fundamental OOP karena menyatukan data dan metode yang relevan menjadi satu unit tunggal (objek), menyembunyikan detail internal, dan hanya menyediakan antarmuka publik yang aman untuk berinteraksi dengan data tersebut, sehingga meningkatkan keamanan data, mempermudah pemeliharaan kode, serta membuat program lebih modular dan tidak kompleks

EMAIL : itsmalik57@gmail.com