Nama: Achmad Gibran
Nim: 202412003

Latihan : Encapsulation & Methods dalam Python

Petunjuk Pengerjaan:

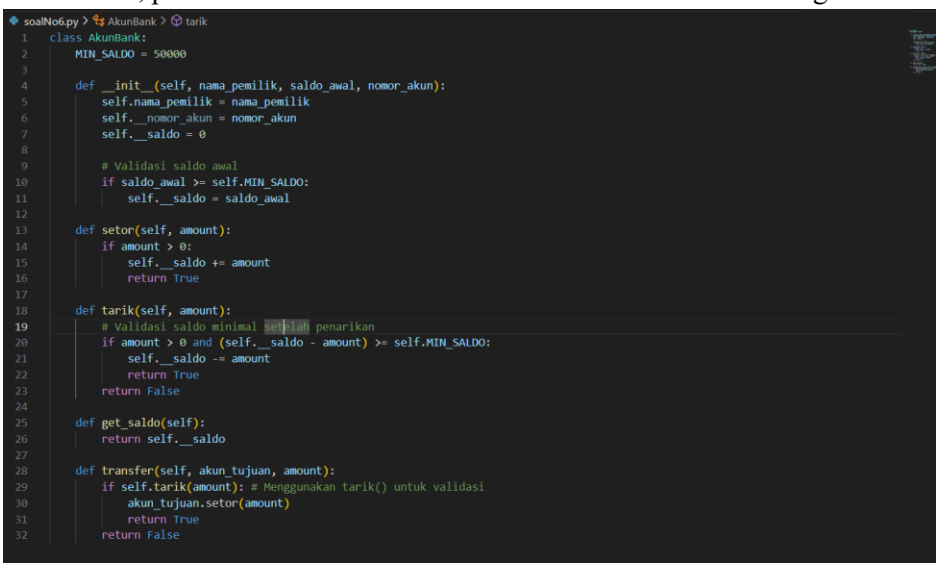
1. Kerjakan soal-soal berikut sesuai dengan pemahaman Anda tentang materi Encapsulation & Methods
2. Untuk soal coding, tuliskan kode Python yang diminta di area yang telah disediakan
3. Gunakan konsep access modifier (public, protected, private) dengan tepat
4. Pastikan untuk menggunakan constructor __init_____dan parameter self dengan benar
5. Anda dapat mengerjakan tugas ini di kertas atau di editor kode favorit Anda

Bagian 1: Soal Teori

1. Jelaskan perbedaan antara instance attributes dan class attributes dalam Python! Berikan contoh sederhana untuk masing-masing.
   - Instance attribute adalah atribut yang melekat pada masing masing objek, sehingga setiap objek memiliki atribut yang berbeda
     Contoh: sama untuk semua mobil, misalnya jumlah_roda = 4 atau merk_pabrikan = "Toyota".
   - Class Attributes adalah atribut yang dimiliki bersama oleh semua objek dan oleh kelas itu sendiri.
     Contoh: berbeda untuk setiap mobil, misalnya warna atau nomor_plat.
2. Apa fungsi dari parameter self dalam metode Python? Mengapa parameter ini diperlukan?
   self itu adalah **objek (benda)** yang sedang dibuat atau diurus. Dia dipakai untuk mengambil data (atribut) dan menjalankan fungsi (*method*) milik objek itu sendiri.
3. Jelaskan perbedaan antara access modifier public, protected, dan private dalam Python! Bagaimana Python menerapkan konsep tersebut?
   - Public (tanpa awalan): Artinya bebas. Siapa pun boleh melihat atau mengubahnya.
   - Protected (satu *underscore*: _nama): Ini hanyalah saran. Seharusnya jangan diakses dari luar, tapi jika mau, Python tidak melarang.
   - Private (dua *underscores*: __nama): Python mengubah namanya (teknik *name mangling*) agar sulit diakses dari luar kelas. Ini untuk menghindari masalah bentrok nama.
4. Apa yang dimaksud dengan "name mangling" dalam Python? Kapan dan mengapa fitur ini digunakan?
   Name Mangling adalah teknik otomatis yang digunakan Python untuk mengubah nama variabel yang didefinisikan dengan awalan dua garis bawah (__nama_variabel) di dalam sebuah kelas.

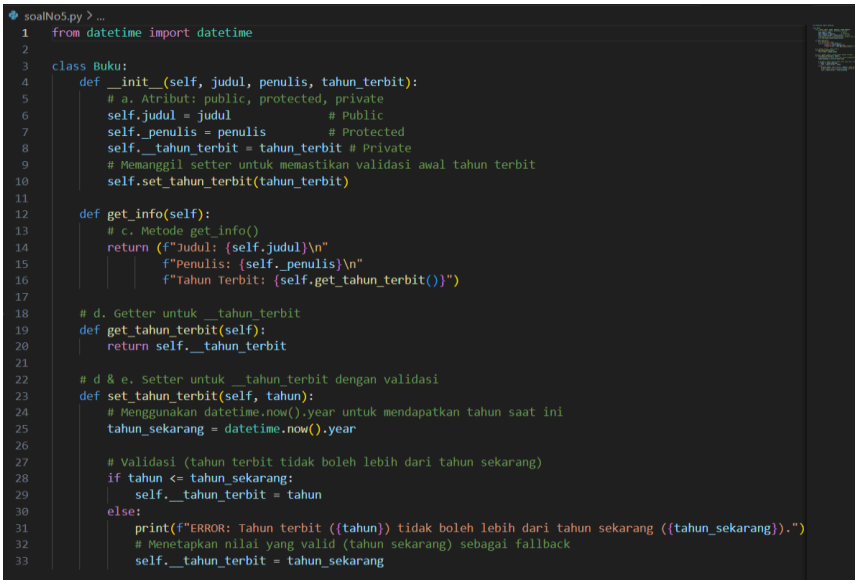
Bagian 2: Soal Coding

5. Buatlah kelas Buku dengan ketentuan berikut:
   a. Memiliki atribut: judul (public), _penulis (protected), __tahun_terbit (private)
   b. Memiliki constructor yang menginisialisasi semua atribut
   c. Memiliki metode get_info() yang mengembalikan string berisi informasi buku
   d. Memiliki metode get_tahun_terbit() (getter) dan set_tahun_terbit(tahun) (setter) untuk mengakses dan mengubah tahun terbit
   e. Pada setter, pastikan tahun terbit tidak boleh lebih dari tahun sekarang



```python
class AkunBank:
    MIN_SALDO = 50000

    def __init__(self, nama_pemilik, saldo_awal, nomor_akun):
        self.nama_pemilik = nama_pemilik
        self.__nomor_akun = nomor_akun
        self.__saldo = 0

        # Validasi saldo awal
        if saldo_awal >= self.MIN_SALDO:
            self.__saldo = saldo_awal

    def setor(self, amount):
        if amount > 0:
            self.__saldo += amount
            return True

    def tarik(self, amount):
        # Validasi saldo minimal setelah penarikan
        if amount > 0 and (self.__saldo - amount) >= self.MIN_SALDO:
            self.__saldo -= amount
            return True
        return False

    def get_saldo(self):
        return self.__saldo

    def transfer(self, akun_tujuan, amount):
        if self.tarik(amount): # Menggunakan tarik() untuk validasi
            akun_tujuan.setor(amount)
            return True
        return False
```

6. Buatlah kelas AkunBank yang menerapkan encapsulation dengan ketentuan:
   a. Memiliki atribut: nama_pemilik (public), __saldo (private), __nomor_akun (private)
   b. Saldo awal harus minimal 50.000 saat pembuatan akun
   c. Memiliki metode: setor(amount), tarik(amount), get_saldo()
   d. Metode tarik harus memastikan saldo tidak boleh kurang dari 50.000
   e. Buat juga metode transfer(akun_tujuan, amount) yang mentransfer dana ke akun lain.



```python
from datetime import datetime

class Buku:
    def __init__(self, judul, penulis, tahun_terbit):
        # a. Atribut: public, protected, private
        self.judul = judul              # Public
        self._penulis = penulis         # Protected
        self.__tahun_terbit = tahun_terbit # Private
        # Memanggil setter untuk memastikan validasi awal tahun terbit
        self.set_tahun_terbit(tahun_terbit)

    def get_info(self):
        # c. Metode get_info()
        return (f"Judul: {self.judul}\n"
                f"Penulis: {self._penulis}\n"
                f"Tahun Terbit: {self.get_tahun_terbit()}")

    # d. Getter untuk __tahun_terbit
    def get_tahun_terbit(self):
        return self.__tahun_terbit

    # d & e. Setter untuk __tahun_terbit dengan validasi
    def set_tahun_terbit(self, tahun):
        # Menggunakan datetime.now().year untuk mendapatkan tahun saat ini
        tahun_sekarang = datetime.now().year

        # Validasi (tahun terbit tidak boleh lebih dari tahun sekarang)
        if tahun <= tahun_sekarang:
            self.__tahun_terbit = tahun
        else:
            print(f"ERROR: Tahun terbit ({tahun}) tidak boleh lebih dari tahun sekarang ({tahun_sekarang}).")
            # Menetapkan nilai yang valid (tahun sekarang) sebagai fallback
            self.__tahun_terbit = tahun_sekarang
```

7. Analisis kode berikut dan jawab pertanyaannya:

```python
class Karyawan:
    perusahaan = "PT Python Jaya" # Class attribute

    def __init__(self, nama, gaji):
        self.nama = nama
        self.__gaji = gaji
        self._tunjangan = 0

    def hitung_gaji_bersih(self):
        return self.__gaji + self._tunjangan

    def set_tunjangan(self, tunjangan):
        if tunjangan >= 0:
            self._tunjangan = tunjangan

# Penggunaan kelas
karyawan1 = Karyawan("Budi", 5000000)
karyawan1.set_tunjangan(1000000)
```

a. Apa yang akan terjadi jika kita mencoba mengakses karyawan1.__gaji langsung dari luar kelas? Mengapa?
Jawab:

```python
# kelas
karyawan1 = Karyawan("Budi", 5000000)
karyawan1.set_tunjangan(1000000)

# Akses langsung atribut _gaji dari luar kelas
print(f"akses langsung ke gaji: {karyawan1._gaji}")
```

```
Traceback (most recent call last):
  File "c:\Users\ACER NITRO\Documents\pemrograman berorientasi objek\kode materi\karyawan.py", line 21, in <module>
    print(f"akses langsung ke gaji: {karyawan1.__gaji}")
                                     ^^^^^^^^^^^^^^^^
AttributeError: 'Karyawan' object has no attribute '__gaji'
```

Jika akses karyawan1.__gaji langsung dari luar kelas , akan terjadi error karena sifatnya private menggunakan dua garis bawah

b. Bagaimana cara yang tepat untuk mengetahui gaji Bud.i?
Jawab:

```python
# memanggil method
gaji_bersih_budi = karyawan1.hitung_gaji_bersih()

# Output
print(f"Gaji Bersih Budi adalah: {gaji_bersih_budi}")
```

Dengan memanggil method yang berisi rumus untuk menghitung gaji bersih keryawan

```
Gaji Bersih Budi adalah: 6000000
```

c. Apakah kita bisa mengubah perusahaan untuk karywan1? Bagaimana caranya?
Bisa, dengan cara menetapkan nilai baru menggunakan nama objek

```python
# mengubah instance
karyawan1.perusahaan = "PT Makmur Jaya"

# cetak
print(karyawan1.perusahaan)
```

```
PT Makmur Jaya
```

d. Mengapa atribut _tunjangan menggunakan underscore tunggal
Karena itu bertujuan memberi tanda kalau atribut _tunjangan tidak seharusnya di akses dari luar, walau aksesnya diijinkan oleh python.