Laporan Praktikum

Pemrograman Berorientasi Objek

Program Studi Teknik Informatika
Institut Teknologi Sumatera
2024

Modul : Abstraksi dan Enkapsulasi

Nama : Rayhan Fadel Irwanto

NIM : 122140236

Kelas (Kelas Asal) : RA

Instruksi sederhana :

- Disarankan kepada **Praktikan Pemrograman Berorientasi Objek** untuk mengeditnya menggunakan Google Docs agar tidak berantakan dan rapi,
- Silahkan mengganti **Nama Modul** baik yang ada pada **Cover** dan **Header** sesuai dengan materi praktikum,
- Gunakan text styling seperti **Heading 1**, **Normal Text** yang telah terformat / Text Style lainnya yang digunakan untuk menjaga estetika laporan,
- Gunakan **Syntax Highlighter** untuk merapikan kode yang sudah Praktikan buat ke dalam Laporan Praktikum.

Materi Praktikum

Dalam pemrograman Python, abstraksi dan enkapsulasi adalah dua konsep yang penting. Abstraksi adalah ide bahwa kita dapat menyembunyikan rincian internal dari sebuah objek dan hanya mengekspos fungsionalitas yang relevan atau penting untuk penggunaan eksternal. Ini membantu dalam mempermudah penggunaan objek tanpa perlu memahami bagaimana implementasinya bekerja di bawah kap. Enkapsulasi, di sisi lain, adalah praktik menyembunyikan rincian internal dari sebuah objek dan hanya membiarkan akses kepadanya melalui antarmuka yang ditentukan. Dalam Python, ini dicapai dengan menggunakan akses atribut private, yang dinyatakan dengan awalan garis bawah ganda (`__`), yang menghalangi akses langsung dari luar kelas. Ini memungkinkan kita untuk menerapkan logika validasi atau proses lain di balik antarmuka publik tanpa perlu khawatir tentang penggunaan yang salah atau manipulasi yang tidak sah dari data. Dengan menerapkan abstraksi dan enkapsulasi dengan benar, kita dapat menciptakan kode yang lebih bersih, lebih mudah dimengerti, dan lebih mudah dipelihara.

Link Source Code

https://onlinegdb.com/XpndxANXu-

Source Code

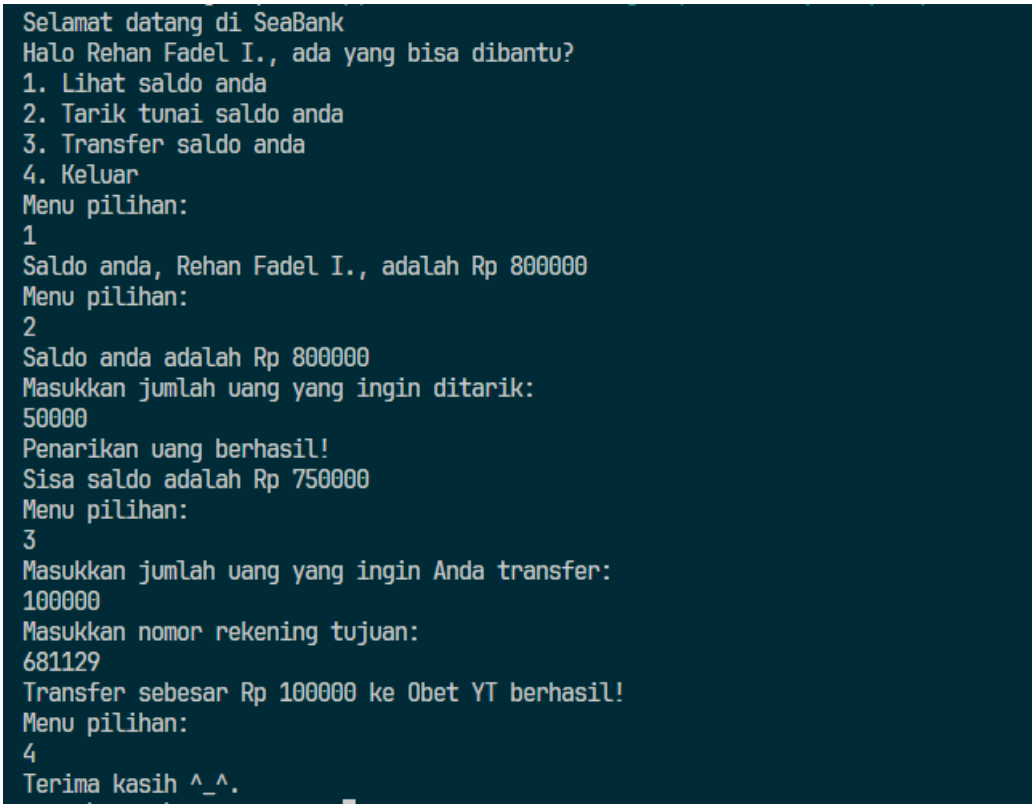
Gunakan **Syntax Highlighter** untuk merapikan Source Code yang dipindahkan dari text editor anda ke dokumen ini.

```
class AkunBank:
    list_pelanggan = []

    def __init__(self, no_pelanggan, nama_pelanggan, jumlah_saldo):
        self.no_pelanggan = no_pelanggan
        self.nama_pelanggan = nama_pelanggan
        self.__jumlah_saldo = jumlah_saldo
        AkunBank.list_pelanggan.append(self)

    def tampilkan_menu(self):
        print("Selamat datang di SeaBank")
        print(f"Halo {self.nama_pelanggan}, ada yang bisa dibantu?")
        print("1. Lihat saldo anda")
        print("2. Tarik tunai saldo anda")
        print("3. Transfer saldo anda")
        print("4. Keluar")
```

```
    def lihat_saldo(self):
        print(f"Saldo anda, {self.nama_pelanggan}, adalah Rp
{self.__jumlah_saldo}")

    def tarik_tunai(self):
        print(f"Saldo anda adalah Rp {self.__jumlah_saldo}")
        print("Masukkan jumlah uang yang ingin ditarik:")
        jumlah_tarik = int(input())
        if jumlah_tarik > self.__jumlah_saldo:
            print("Maaf, saldo Anda tidak mencukupi untuk melakukan
penarikan.")
        else:
            self.__jumlah_saldo -= jumlah_tarik
            print("Penarikan uang berhasil!")
            print(f"Sisa saldo adalah Rp {self.__jumlah_saldo}")


    def transfer(self):
        print("Masukkan jumlah uang yang ingin Anda transfer:")
        nominal_transfer = int(input())
        print("Masukkan nomor rekening tujuan:")
        no_rekening_tujuan = int(input())
        tujuan = None
        for akun in AkunBank.list_pelanggan:
            if akun.no_pelanggan == no_rekening_tujuan:
                tujuan = akun
                break
        if tujuan is None:
            print("Nomor rekening tujuan tidak ditemukan dalam sistem kami.")
        else:
            if self.__jumlah_saldo >= nominal_transfer:
                self.__jumlah_saldo -= nominal_transfer
                tujuan.__jumlah_saldo += nominal_transfer
                print(f"Transfer sebesar Rp {nominal_transfer} ke
{tujuan.nama_pelanggan} berhasil!")
            else:
                print("Maaf, saldo Anda tidak mencukupi untuk melakukan
transfer.")

# Mendeklarasi Object dari class AkunBank
AkunBank1 = AkunBank(5050505, "Rehan Fadel I.", 800000)
AkunBank2 = AkunBank(681129, "Obet YT", 17000)
AkunBank3 = AkunBank(19328480, "Raihan Ekspor", 90000000)

# Simulasi penggunaan fungsi-fungsi
```

```
AkunBank1.tampilkan_menu()

while True:
    print("Menu pilihan:")
    pilihan = int(input())
    if pilihan == 1:
        AkunBank1.lihat_saldo()
    elif pilihan == 2:
        AkunBank1.tarik_tunai()
    elif pilihan == 3:
        AkunBank1.transfer()
    elif pilihan == 4:
        print("Terima kasih ^_^.")
        break
```

Dokumentasi Hasil Running



Gambar 1. Output Studi Kasus Abstraksi dan Enkapsulasi pada Class AkunBank

Kode di atas mengimplementasikan sebuah kelas `AkunBank` yang memungkinkan pengguna untuk melihat saldo, melakukan penarikan tunai, dan mentransfer dana. Setiap objek `AkunBank` memiliki atribut nomor pelanggan (`no_pelanggan`), nama pelanggan (`nama_pelanggan`), dan saldo (`__jumlah_saldo`). Fungsi `tampilkan_menu()` digunakan untuk menampilkan menu layanan bank kepada pelanggan. Fungsi `lihat_saldo()` memungkinkan pelanggan untuk melihat saldo mereka, sedangkan fungsi `tarik_tunai()` memungkinkan mereka untuk menarik uang dari akun mereka dengan validasi saldo yang cukup. Fungsi `transfer()` memungkinkan pelanggan untuk mentransfer dana ke akun lain dengan memeriksa saldo yang mencukupi. Dalam simulasi, pengguna dapat memilih menu untuk melakukan operasi yang diinginkan, seperti melihat saldo, menarik tunai, atau mentransfer dana, hingga mereka memilih untuk keluar dari layanan bank.

Tugas 2 Demonstrasi perbedaan atribut/fungsi private dan publik

Link Source Code

https://onlinegdb.com/PxOC8ajG6

Source Code

```
class Produk:
    # Atribut private, hanya dapat diakses dari dalam kelas
    __kode_produk = "P001"
    __harga = 0

    # Atribut protected, dapat diakses dari dalam kelas dan kelas turunannya
    _stok = 0

    # Atribut public, dapat diakses dari luar kelas
    nama_produk = "Produk ABC"

    def __init__(self, nama):
        self.nama = nama

    # Metode private, hanya dapat dipanggil dari dalam kelas
    def __update_harga(self, harga_baru):
        self.__harga = harga_baru
        print("Harga produk berhasil diperbarui")

    # Metode public, dapat dipanggil dari luar kelas
    def tampilkan_info(self):
        print(f"Nama Produk: {self.nama}")
        print(f"Kode Produk: {self.__kode_produk}")
        print(f"Harga: Rp {self.__harga}")
        print(f"Stok: {self._stok}")

    # Metode untuk mengubah harga produk, dapat dipanggil dari luar kelas
    def ubah_harga(self, harga_baru):
        self.__update_harga(harga_baru)


# Membuat objek produk
produk1 = Produk("Laptop Legion")

# Mengakses atribut public dari luar kelas
print("Nama produk:", produk1.nama_produk)

# Mengakses atribut protected dari luar kelas
```

```
print("Stok produk:", produk1._stok)

# Percobaan mengakses atribut private dari luar kelas akan menghasilkan error
# print("Kode produk:", produk1.__kode_produk)  # Error: AttributeError

# Memanggil metode public untuk menampilkan informasi produk
produk1.tampilkan_info()

# Percobaan memanggil metode private dari luar kelas akan menghasilkan error
# produk1.__update_harga(15000000)  # Error: AttributeError

# Memanggil metode public untuk mengubah harga produk
produk1.ubah_harga(15000000)

# Memanggil metode public untuk menampilkan informasi produk setelah mengubah
harga
produk1.tampilkan_info()
```

Dokumentasi Hasil Running

```
Nama produk: Produk ABC
Stok produk: 0
Nama Produk: Laptop Legion
Kode Produk: P001
Harga: Rp 0
Stok: 0
Harga produk berhasil diperbarui
Nama Produk: Laptop Legion
Kode Produk: P001
Harga: Rp 15000000
Stok: 0
```

Gambar 2. Output Penjelasan Perbedaan Atrribut/Fungsi Privat dan Publik

Pada kode di atas, terdapat kelas `Produk` yang memiliki atribut dan metode dengan berbagai tingkat aksesibilitas. Atribut `__kode_produk` dan `__harga` bersifat privat, yang berarti hanya dapat diakses dari dalam kelas `Produk` saja. Di sisi lain, atribut `nama_produk` adalah publik, sehingga dapat diakses dari luar kelas. Metode `__update_harga()` juga bersifat privat, hanya dapat dipanggil dari dalam kelas `Produk`, sedangkan metode `tampilkan_info()` dan `ubah_harga()` adalah publik, sehingga dapat dipanggil dari luar kelas.

Ini menunjukkan perbedaan dalam tingkat aksesibilitas antara atribut dan metode privat yang hanya dapat diakses secara internal dengan atribut dan metode publik yang dapat diakses dari luar kelas.