LAPORAN PRAKTIKUM PEMROGRAMAN BERORIENTASI OBJEK (PBO)

PRAKTIKUM 10

2411102441126

Jauzah Chalifa Chairunnisa

FAKULTAS SAINS DAN TEKNOLOGI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

EMAIL : 2411102441126@gmail.com

PRAKTIKUM: Penerapan pada Sistem Perpustakaan

Pada praktikum ini, konsep Pemrograman Berorientasi Objek (PBO) diterapkan untuk membuat model sederhana dari sistem perpustakaan digital. Tujuan utamanya adalah memahami bagaimana objek-objek dalam dunia nyata (buku, anggota, dan perpustakaan) dapat diterjemahkan menjadi kelas dan objek dalam program.

Struktur Kelas yang Digunakan

1. Kelas Buku
   Kelas ini mewakili setiap buku yang ada di perpustakaan.
   Di dalamnya terdapat atribut seperti:
   - judul → nama buku
   - penulis → nama pengarang buku
   - dipinjam → status buku (True jika sedang dipinjam, False jika tersedia)

Kelas ini juga memiliki fungsi untuk menandai buku saat dipinjam atau dikembalikan.

2. Kelas Pustaka
   Kelas ini berperan sebagai pengelola seluruh koleksi buku.
   Ia menyimpan daftar semua objek Buku yang ada di perpustakaan dan menyediakan fungsi untuk:
   - Menambahkan buku baru ke katalog
   - Menampilkan seluruh daftar buku beserta statusnya (tersedia atau dipinjam)

Dengan cara ini, kita bisa melihat kondisi seluruh koleksi secara langsung.

3. Kelas Anggota
   Kelas ini menggambarkan orang yang menjadi pengguna perpustakaan.
   Setiap anggota dapat:
   - Meminjam buku: jika buku masih tersedia, maka statusnya diubah menjadi *dipinjam*, dan buku tersebut masuk ke daftar pinjaman anggota.
   - Mengembalikan buku: ketika buku dikembalikan, statusnya kembali menjadi *tersedia* dan dihapus dari daftar pinjaman anggota.

Proses ini menunjukkan bagaimana objek Anggota berinteraksi langsung dengan objek Buku.
Inti dari Praktikum
Melalui percobaan ini, kita dapat memahami hubungan antarobjek dalam PBO:

- Objek Anggota bisa mengubah status pada objek Buku melalui aksi pinjam dan kembalikan.
- Objek Pustaka berfungsi sebagai pengawas, yang dapat melaporkan seluruh perubahan status buku di dalam sistem.

Dengan cara ini, sistem perpustakaan menjadi lebih realistis, karena setiap objek memiliki peran dan tanggung jawab masing-masing, mirip seperti dunia nyata.

EMAIL : 2411102441126@gmail.com

pertemuan_10.py

```python
# Definisi class Buku
class Buku:
    def __init__(self, judul, penulis):
        self.judul = judul
        self.penulis = penulis
        self.dipinjam = False

    def tampilkan_info(self):
        status = "Dipinjam" if self.dipinjam else "Tersedia"
        return f"Buku: {self.judul}, Penulis: {self.penulis}, Status: {status}"


# Definisi class Pustaka
class Pustaka:
    def __init__(self):
        self.koleksi_buku = []

    def tambah_buku(self, buku):
        self.koleksi_buku.append(buku)

    def tampilkan_koleksi(self):
        for buku in self.koleksi_buku:
            print(buku.tampilkan_info())


# Definisi class Anggota
class Anggota:
    def __init__(self, nama):
        self.nama = nama
        self.daftar_buku = []

    def pinjam_buku(self, buku):
        if not buku.dipinjam:
            buku.dipinjam = True
            self.daftar_buku.append(buku)
            print(f"{self.nama} meminjam buku '{buku.judul}'")
        else:
            print(f"Buku '{buku.judul}' sedang dipinjam orang lain.")

    def kembalikan_buku(self, buku):
        if buku in self.daftar_buku:
            buku.dipinjam = False
            self.daftar_buku.remove(buku)
            print(f"{self.nama} mengembalikan buku '{buku.judul}'")
        else:
```

EMAIL : 2411102441126@gmail.com

pertemuan_10.py > ...

```
            print(buku.tampilkan_info())


# Definisi class Anggota
class Anggota:
    def __init__(self, nama):
        self.nama = nama
        self.daftar_buku = []

    def pinjam_buku(self, buku):
        if not buku.dipinjam:
            buku.dipinjam = True
            self.daftar_buku.append(buku)
            print(f"{self.nama} meminjam buku '{buku.judul}'")
        else:
            print(f"Buku '{buku.judul}' sedang dipinjam orang lain.")

    def kembalikan_buku(self, buku):
        if buku in self.daftar_buku:
            buku.dipinjam = False
            self.daftar_buku.remove(buku)
            print(f"{self.nama} mengembalikan buku '{buku.judul}'")
        else:
            print(f"{self.nama} tidak meminjam buku '{buku.judul}'")


# --- Membuat objek dan menjalankan simulasi ---
buku1 = Buku("Harry Potter", "J.K. Rowling")
buku2 = Buku("The Hobbit", "J.R.R. Tolkien")

pustaka = Pustaka()
pustaka.tambah_buku(buku1)
pustaka.tambah_buku(buku2)

anggota1 = Anggota("Andi")
anggota1.pinjam_buku(buku1)
anggota1.pinjam_buku(buku2)

print("\nStatus Koleksi Pustaka:")
pustaka.tampilkan_koleksi()

anggota1.kembalikan_buku(buku1)
print("\nStatus Koleksi Setelah Pengembalian:")
pustaka.tampilkan_koleksi()
```

EMAIL : 2411102441126@gmail.com

Output

```
PS D:\TUGAS TI\TUGAS SMT 3\PEMOGRAMAN WEB\CODINGGG\P10> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "d:/TUGAS TI/TUGAS SMT 3/PEMOGRAMAN WEB/CODINGGG/P10/pertemuan_10.py"
Andi meminjam buku 'Harry Potter'
Andi meminjam buku 'The Hobbit'

Status Koleksi Pustaka:
Buku: Harry Potter, Penulis: J.K. Rowling, Status: Dipinjam
Buku: The Hobbit, Penulis: J.R.R. Tolkien, Status: Dipinjam
Andi mengembalikan buku 'Harry Potter'

Status Koleksi Setelah Pengembalian:
Buku: Harry Potter, Penulis: J.K. Rowling, Status: Tersedia
Buku: The Hobbit, Penulis: J.R.R. Tolkien, Status: Dipinjam
PS D:\TUGAS TI\TUGAS SMT 3\PEMOGRAMAN WEB\CODINGGG\P10> & C:/Users/ASUS/AppData/Local/Programs/Python/Python312/python.exe "d:/TUGAS TI/TUGAS SMT 3/PEMOGRAMAN WEB/CODINGGG/P10/pertemuan_10.py"
Andi meminjam buku 'Harry Potter'
Andi meminjam buku 'The Hobbit'

Status Koleksi Pustaka:
Buku: Harry Potter, Penulis: J.K. Rowling, Status: Dipinjam
Buku: The Hobbit, Penulis: J.R.R. Tolkien, Status: Dipinjam
Andi mengembalikan buku 'Harry Potter'

Status Koleksi Setelah Pengembalian:
Buku: Harry Potter, Penulis: J.K. Rowling, Status: Tersedia
Buku: The Hobbit, Penulis: J.R.R. Tolkien, Status: Dipinjam
PS D:\TUGAS TI\TUGAS SMT 3\PEMOGRAMAN WEB\CODINGGG\P10>
```

Studi Kasus: Sistem Toko

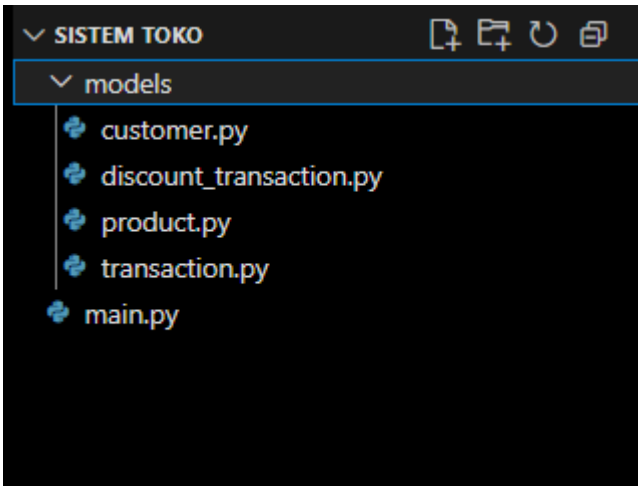
Sistem toko ini menunjukkan bagaimana konsep **OOP** digunakan untuk membangun proses jual beli sederhana.
Terdiri dari empat kelas utama:

1. **Product** – menyimpan data barang (nama, harga, stok) dan dapat mengurangi stok saat dibeli.
2. **Customer** – mewakili pelanggan yang dapat menambahkan produk ke keranjang dan melihat total belanja.
3. **Transaction** – memproses pembelian, menghitung total harga, dan mengurangi stok produk.
4. **DiscountTransaction** – turunan dari Transaction dengan tambahan fitur diskon.

Melalui interaksi antarobjek, sistem dapat mencatat pembelian pelanggan, menghitung total, dan menerapkan diskon jika berlaku.
Penerapan OOP membuat sistem **lebih rapi, fleksibel, dan mudah dikembangkan**.

Konsep OOP yang Diterapkan

Konsep	Penjelasan
Encapsulation	Atribut seperti `nama_produk`, `harga`, dan `stok` dibuat privat agar tidak bisa diakses langsung dari luar class. Akses dilakukan melalui *getter* dan *setter* untuk menjaga keamanan data.
Inheritance	Class `DiscountTransaction` merupakan turunan dari `Transaction` yang mewarisi atribut dan method-nya, lalu menambahkan fitur diskon agar tidak perlu menulis ulang kode.
Polymorphism	Method `process()` di-override pada `DiscountTransaction` untuk menambahkan logika perhitungan diskon tanpa mengubah class induk.
Modularization	Setiap class disimpan dalam file terpisah (`product.py`, `customer.py`, `transaction.py`, `discount_transaction.py`) agar kode rapi, mudah dibaca, dan mudah dikelola.



EMAIL : 2411102441126@gmail.com

Code dan Penjelasan

- Init.py

File ini berisi class Produk dan class turunan ProdukElektronik.

- Produk adalah *parent class* yang menyimpan data dasar barang: nama, harga, dan stok.
- Ada fungsi untuk melihat dan mengubah data (getter & setter), serta menambah atau mengurangi stok.
- Fungsi tampilkan_info() menampilkan detail produk.

Kemudian ada ProdukElektronik sebagai *child class* yang mewarisi semua atribut dan fungsi dari Produk, tapi menambah atribut baru yaitu garansi.

- Fungsi tampilkan_info() di-*override* agar juga menampilkan lama garansi.

Jadi, file ini menunjukkan konsep OOP dasar: *enkapsulasi, inheritance,* dan *polymorphism*.

init.py > ...

```python
# models/produk.py

# Parent Class
class Produk:
    def __init__(self, nama, harga, stok):
        self.__nama = nama
        self.__harga = harga
        self.__stok = stok

    # Getter dan Setter (enkapsulasi)
    def get_nama(self):
        return self.__nama

    def get_harga(self):
        return self.__harga

    def get_stok(self):
        return self.__stok

    def set_harga(self, harga_baru):
        if harga_baru > 0:
            self.__harga = harga_baru
        else:
            print("Harga tidak valid!")

    def kurangi_stok(self, jumlah):
        if jumlah <= self.__stok:
            self.__stok -= jumlah
        else:
            print(f"Stok {self.__nama} tidak mencukupi.")

    def tambah_stok(self, jumlah):
        self.__stok += jumlah

    def tampilkan_info(self):
        return f"Produk: {self.__nama}, Harga: Rp{self.__harga}, Stok: {self.__stok}"


# Child Class (Inheritance & Polymorphism)
class ProdukElektronik(Produk):
    def __init__(self, nama, harga, stok, garansi):
        super().__init__(nama, harga, stok)
        self.garansi = garansi

    def tampilkan_info(self):  # Polymorphism
        return f"{super().tampilkan_info()}, Garansi: {self.garansi} tahun"

```

EMAIL : 2411102441126@gmail.com

- Produk.py

File ini berisi kelas Produk sebagai induk dan ProdukElektronik sebagai turunannya.

- Produk menyimpan data dasar barang seperti nama, harga, dan stok. Di dalamnya ada fungsi untuk:
  - Melihat data produk (getter).
  - Mengubah harga (setter).
  - Menambah atau mengurangi stok.
  - Menampilkan informasi produk.

    Ini menunjukkan konsep enkapsulasi, karena atribut dibuat privat (__nama, __harga, __stok) dan hanya bisa diakses lewat fungsi.
- ProdukElektronik adalah turunan dari Produk, jadi mewarisi semua fungsinya, tapi menambahkan atribut baru yaitu garansi.

  Fungsi tampilkan_info() di-*override* agar menampilkan juga lama garansi.

Jadi intinya:

File ini menerapkan konsep **OOP** (Pemrograman Berorientasi Objek) dengan **enkapsulasi**, **inheritance**, dan **polymorphism**.

produk.py > ...

```python
# models/produk.py

# Parent Class
class Produk:
    def __init__(self, nama, harga, stok):
        self.__nama = nama
        self.__harga = harga
        self.__stok = stok

    # Getter dan Setter (enkapsulasi)
    def get_nama(self):
        return self.__nama

    def get_harga(self):
        return self.__harga

    def get_stok(self):
        return self.__stok

    def set_harga(self, harga_baru):
        if harga_baru > 0:
            self.__harga = harga_baru
        else:
            print("Harga tidak valid!")

    def kurangi_stok(self, jumlah):
        if jumlah <= self.__stok:
            self.__stok -= jumlah
        else:
            print(f"Stok {self.__nama} tidak mencukupi.")

    def tambah_stok(self, jumlah):
        self.__stok += jumlah

    def tampilkan_info(self):
        return f"Produk: {self.__nama}, Harga: Rp{self.__harga}, Stok: {self.__stok}"


# Child Class (Inheritance & Polymorphism)
class ProdukElektronik(Produk):
    def __init__(self, nama, harga, stok, garansi):
        super().__init__(nama, harga, stok)
        self.garansi = garansi

    def tampilkan_info(self):  # Polymorphism
        return f"{super().tampilkan_info()}, Garansi: {self.garansi} tahun"

```

EMAIL : 2411102441126@gmail.com

- Pelanggan.py

File ini berisi kelas Pelanggan yang mewakili pembeli di sistem toko.

- Saat objek dibuat, pelanggan punya nama dan keranjang belanja (list kosong).
- Fungsi tambah_ke_keranjang() digunakan untuk menambahkan produk ke keranjang:
  - Mengecek dulu apakah stok produk cukup.
  - Kalau cukup, produk dan jumlahnya dimasukkan ke keranjang.
  - Kalau stok tidak cukup, muncul pesan peringatan.
- Fungsi tampilkan_keranjang() menampilkan isi keranjang pelanggan, yaitu daftar produk dan jumlah yang dibeli.

- Transaksi.py

File ini berisi kelas Transaksi, yang berfungsi menangani proses pembelian pelanggan.

- Saat objek dibuat, transaksi menyimpan data pelanggan dan mengatur total pembayaran mulai dari nol.
- Fungsi proses_pembelian() melakukan langkah-langkah berikut:
    1. Menampilkan pesan bahwa transaksi sedang diproses.
    2. Mengambil setiap produk di keranjang pelanggan.
    3. Mengurangi stok produk sesuai jumlah yang dibeli.
    4. Menghitung subtotal (harga × jumlah) dan menambahkannya ke total.
    5. Menampilkan rincian harga tiap produk dan total pembayaran.
    6. Setelah selesai, keranjang pelanggan dikosongkan.

Jadi, kelas ini bertanggung jawab untuk menghitung total belanja, mengurangi stok produk, dan menyelesaikan transaksi pelanggan.

transaksi.py > ...

```python
# models/transaksi.py

class Transaksi:
    def __init__(self, pelanggan):
        self.pelanggan = pelanggan
        self.total = 0

    def proses_pembelian(self):
        print(f"\nMemproses transaksi untuk {self.pelanggan.nama}...")
        for produk, jumlah in self.pelanggan.keranjang:
            produk.kurangi_stok(jumlah)
            subtotal = produk.get_harga() * jumlah
            self.total += subtotal
            print(f"{produk.get_nama()} x{jumlah} = Rp{subtotal}")
        print(f"Total Pembayaran: Rp{self.total}")
        self.pelanggan.keranjang.clear()

```

- Main.py

File ini adalah program utama (main.py) yang menjalankan keseluruhan sistem toko.

1. Mengimpor kelas-kelas penting dari file lain — yaitu Produk, ProdukElektronik, Pelanggan, dan Transaksi.
2. Membuat objek produk:
    - Laptop Asus, HP Samsung, dan Buku Python, masing-masing dengan harga dan stok tertentu.
3. Menampilkan informasi produk ke layar (nama, harga, dan stok).
4. Membuat pelanggan bernama *Icha*.
5. Menambahkan produk ke keranjang — misalnya Icha membeli 1 laptop dan 2 buku.
6. Menampilkan isi keranjang agar terlihat barang apa saja yang akan dibeli.
7. Memproses transaksi — sistem menghitung total harga, mengurangi stok produk, dan menampilkan total pembayaran.
8. Menampilkan stok produk setelah pembelian, untuk memastikan stok berkurang sesuai jumlah yang dibeli.

Jadi secara sederhana: file ini menggabungkan semua komponen (produk, pelanggan, transaksi) untuk mensimulasikan proses belanja di toko dari awal sampai akhir.

main.py 3 | python main.py.py 3 | transaksi.py

D: > TUGAS TI > TUGAS SMT 3 > PBO > praktikum > P 10 > sistem toko > python main.py.py > ...

```python
from models.produk import Produk, ProdukElektronik
from models.pelanggan import Pelanggan
from models.transaksi import Transaksi

if __name__ == "__main__":
    # Membuat produk
    laptop = ProdukElektronik("Laptop Asus", 9000000, 5, 2)
    hp = ProdukElektronik("HP Samsung", 4000000, 10, 1)
    buku = Produk("Buku Python", 150000, 20)

    # Menampilkan info produk
    print(laptop.tampilkan_info())
    print(hp.tampilkan_info())
    print(buku.tampilkan_info())

    # Membuat pelanggan
    pelanggan1 = Pelanggan("Icha")

    # Pelanggan menambahkan produk ke keranjang
    pelanggan1.tambah_ke_keranjang(laptop, 1)
    pelanggan1.tambah_ke_keranjang(buku, 2)
    pelanggan1.tampilkan_keranjang()

    # Proses transaksi
    transaksi1 = Transaksi(pelanggan1)
    transaksi1.proses_pembelian()

    # Cek stok setelah transaksi
    print("\nStok setelah pembelian:")
    print(laptop.tampilkan_info())
    print(buku.tampilkan_info())

```

Output:

```
PS D:\TUGAS TI\TUGAS SMT 3\PBO\praktikum\P 10\sistem toko\models> & C:/Users/A
:/TUGAS TI/TUGAS SMT 3/PBO/praktikum/P 10/sistem toko/python main.py.py"
Produk: Laptop Asus, Harga: Rp9000000, Stok: 5, Garansi: 2 tahun
Produk: HP Samsung, Harga: Rp4000000, Stok: 10, Garansi: 1 tahun
Produk: Buku Python, Harga: Rp150000, Stok: 20
Icha menambahkan 1 Laptop Asus ke keranjang.
Icha menambahkan 2 Buku Python ke keranjang.

Keranjang Icha:
- Laptop Asus x1
- Buku Python x2

Memproses transaksi untuk Icha...
Laptop Asus x1 = Rp9000000
Buku Python x2 = Rp300000
Total Pembayaran: Rp9300000

Stok setelah pembelian:
Produk: Laptop Asus, Harga: Rp9000000, Stok: 4, Garansi: 2 tahun
Produk: Buku Python, Harga: Rp150000, Stok: 18
PS D:\TUGAS TI\TUGAS SMT 3\PBO\praktikum\P 10\sistem toko\models>
```

EMAIL : 2411102441126@gmail.com

Kesimpulan:

Secara ringkas:

1. **Setiap objek memiliki peran dan tanggung jawab sendiri** misalnya Buku di perpustakaan atau Produk di toko.
2. **Objek dapat saling berinteraksi** seperti Anggota meminjam Buku, atau Pelanggan membeli Produk.
3. **Konsep utama OOP diterapkan secara nyata:**
    - **Encapsulation (enkapsulasi):** Data penting disembunyikan dan hanya bisa diakses lewat metode khusus (getter & setter).
    - **Inheritance (pewarisan):** Kelas turunan seperti ProdukElektronik mewarisi fungsi dari Produk, lalu menambah fitur baru tanpa menulis ulang kode.
    - **Polymorphism (polimorfisme):** Metode seperti tampilkan_info() diubah perilakunya pada kelas turunan agar menampilkan informasi berbeda.
    - **Modularization (modularisasi):** Setiap kelas disimpan di file terpisah, membuat program lebih mudah dibaca dan dikelola.

Kesimpulan akhir:

Melalui penerapan OOP, sistem menjadi **lebih terorganisir, mudah dikembangkan, dan mencerminkan interaksi dunia nyata** baik untuk sistem perpustakaan maupun sistem toko digital.

EMAIL : 2411102441126@gmail.com