LAPORAN PRAKTIKUM
Object dan Class (Enkapsulasi)

Disusun Oleh:
Dzakir Tsabit Asy Syafiq (241511071)
Jurusan Teknik Komputer dan Informatika

Program Studi D-3 Teknik Informatika
Politeknik Negeri Bandung
28/08/2025

APLIKASI MEMBUAT MENU PRODUK PADA KOPERASI

1. Deskripsi :

1. **Menampilkan Daftar Produk**
    - Program menyimpan data produk dalam bentuk **objek dari class Produk**.
    - Setiap produk memiliki atribut **nama**, **harga**, dan **stok**.
    - Daftar produk ditampilkan secara otomatis saat program dijalankan, dengan format:

    1. Pulpen - Rp 3500

    2. Buku - Rp 7000

    3. Pensil - Rp 300

    - Nomor urut digunakan agar pengguna bisa memilih produk dengan mudah.
2. **Memilih Produk**
    - Pengguna diminta untuk memasukkan angka sesuai nomor produk yang ditampilkan.
    - Program melakukan **validasi input**:
        - Jika nomor yang dimasukkan kurang dari 1 atau lebih besar dari jumlah produk, maka akan muncul pesan **"Pilihan tidak valid"** dan program berhenti.
        - Jika input valid, maka program mengambil data produk sesuai pilihan pengguna.
3. **Menginput Jumlah Barang**
    - Setelah memilih produk, pengguna diminta untuk menginput jumlah barang yang ingin dibeli.
    - Jumlah barang yang dimasukkan akan disimpan dalam sebuah **objek transaksi**.
    - (Opsional) Program bisa memeriksa apakah jumlah barang melebihi stok. Jika iya, maka transaksi ditolak.
4. **Menghitung Total Harga**
    - Program menghitung total harga dengan rumus:

    total = harga_satuan × jumlah_barang

    - Perhitungan dilakukan otomatis setelah pengguna memasukkan jumlah barang.
5. **Menampilkan Struk Pembelian**

- Setelah transaksi dilakukan, program mencetak struk pembelian secara otomatis.
- Struk menampilkan informasi:
  - Nama produk yang dibeli
  - Harga satuan produk
  - Jumlah barang yang dibeli
  - Total harga yang harus dibayar

Contoh format struk:

-- STRUK KOPERASI --

Nama Barang : Buku

Harga Satuan : 7000

Jumlah Beli : 2

Total Bayar : 14000

---

6. **Struktur Program**
   - **Class Produk**: menyimpan data produk (nama, harga, stok).
   - **Class Transaksi**: mengatur proses pembelian (jumlah beli, total harga, struk).
   - **Class koperasi (main class)**: menampilkan menu, menerima input pengguna, dan memproses transaksi.

2. Source Code :

File name Produk.java (Object & Class 1):

```
public class Produk {

private String nama_prod;

private int harga_prod;

private int stok_prod;

private int jumlah_prod;


public Produk(String nama_prod, int harga_prod,int stok_prod,int jumlah_prod)
{

this.nama_prod = nama_prod;

this.harga_prod = harga_prod;
```

```
this.stok_prod = stok_prod;
this.jumlah_prod = jumlah_prod;
}
public String Getnama_prod(){
return nama_prod;
}
public int Getharga_prod(){
return harga_prod;
}
public int Getstok_prod(){
return stok_prod;
}
public int Getjumlah_prod(){
return jumlah_prod;
}

}
```

File name Transaksi.java (Object & Class 2) :

```
public class Transaksi {
    private Produk produk;
    private int jumlahBeli;

    public Transaksi(Produk produk,int jumlahBeli){
        this.produk = produk;
        this.jumlahBeli = jumlahBeli;
    }

    public Produk getProd(){
        return produk;
    }

    public int GetjumlahBeli(){
        return jumlahBeli;
    }
```

```
    public int hitungTotal() {
        return produk.Getharga_prod() * jumlahBeli;
    }

    public void cetak_struk(){
        System.out.println("-- STRUK KOPERASI --");
        System.out.println("Nama barang: " + produk.Getnama_prod());
        System.out.println("Harga satuan: " + produk.Getharga_prod());
        System.out.println("Jumlah beli: " + jumlahBeli);
        System.out.println("Total bayar: " + hitungTotal());
        System.out.println("--------------------");
    }
}
```

File name koperasi.java (Main) :

```
import java.util.Scanner;

public class koperasi{
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // array produk, bisa di tambahkan disini
        Produk[] daftarProduk = {
            new Produk("Pulpen", 3500,10, 0),
            new Produk("Buku",7000,15,0),
            new Produk("Pensil",300,7,0)
        };

        // looping menu langsung menampilkan array nya
        System.out.println("-- MENU PRODUK KOEPARSI --");
        for (int i = 0 ;i < daftarProduk.length; i++) {
            System.out.println((i+1) + ". " + daftarProduk[i].Getnama_prod() + " Rp.
" + daftarProduk[i].Getharga_prod()  );
        }
```

```
        // input dengan berdasarkan daftar produk length
        System.out.print("Pilih barang yang mau dibeli (1 - " + daftarProduk.length +
"): " );
        int pilih = input.nextInt();

        //error handling
        if (pilih < 1 || pilih > daftarProduk.length){
            System.out.println("Pilihan tidak valid..");
            return;
        }

        Produk produkDipilih = daftarProduk[pilih - 1];

        System.out.print("Jumlah barang yang ingin dibeli: ");
        int jumlah = input.nextInt();

        Transaksi trx = new Transaksi(produkDipilih,jumlah);

        trx.cetak_struk();

    }
}
```

3. Hasil Output :

```
-- MENU PRODUK KOEPARSI --
1. Pulpen Rp. 3500
2. Buku Rp. 7000
3. Pensil Rp. 300
Pilih barang yang mau dibeli (1 - 3): 2
Jumlah barang yang ingin dibeli: 2
-- STRUK KOPERASI --
Nama barang: Buku
Harga satuan: 7000
Jumlah beli: 2
Total bayar: 14000
--------------------
```

4. Lesson Learn :

- **Pentingnya Pemodelan dengan Class**
    - Dengan membuat class Produk, data setiap barang bisa disimpan lebih rapi dalam bentuk objek.
    - Menambahkan class kedua seperti Transaksi membantu memisahkan tanggung jawab (produk hanya menyimpan data barang, sedangkan transaksi mengatur pembelian dan perhitungan).
    - Hal ini menunjukkan konsep **Object-Oriented Programming (OOP)** yaitu **Encapsulation** dan **Separation of Concern**.
- **Manfaat Array of Object**
    - Dengan menggunakan Produk[] daftarProduk, kita bisa menyimpan banyak produk dalam satu struktur data.
    - Hal ini memudahkan penambahan produk baru tanpa harus menulis kode yang berulang.
- **Validasi Input Itu Penting**
    - Program harus menangani input yang salah (misalnya memilih nomor produk yang tidak ada).
    - Dengan validasi, program lebih **robust** dan tidak mudah error saat dijalankan oleh pengguna.
- **Interaksi dengan Pengguna Lebih Jelas**
    - Menampilkan daftar produk dalam format menu mempermudah pengguna memilih produk.
    - Adanya struk pembelian membuat output lebih informatif dan menyerupai proses nyata.
- **Perhitungan Otomatis Mengurangi Kesalahan**
    - Dengan membuat fungsi hitungTotal() di class Transaksi, perhitungan harga menjadi otomatis.
    - Ini menunjukkan pentingnya **membuat metode khusus untuk perhitungan**, agar kode lebih terstruktur dan mudah dipelihara.
- **Penerapan Konsep Reusable Code**
    - Metode seperti getProduk(), getJumlahBeli(), dan hitungTotal() dapat dipakai ulang di bagian lain program.
    - Hal ini mendukung prinsip **DRY (Don’t Repeat Yourself)**.
- **Pentingnya Perencanaan Struktur Program**

- Dari pengalaman ini, terlihat bahwa dengan menambahkan class kedua (Transaksi), program menjadi lebih mudah dikembangkan ke fitur lanjutan (contoh: update stok, diskon, transaksi multi-produk).
- Ini membuktikan bahwa **desain awal sangat memengaruhi kemudahan pengembangan program di masa depan**.