**LAPORAN PRAKTIKUM**

**Implementasi OOP: Package, Static, Dependency, dan Aggregation pada Aplikasi Koperasi**



**Disusun Oleh:**

**Dzakir Tsabit Asy Syafiq (241511071)**

**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika**

**Politeknik Negeri Bandung**

**06/09/2025**

**1. PENERAPAN STATIC DAN PACKAGE**

**1.1 Struktur Package**

Aplikasi ini mengorganisasi kode dalam dua package utama:

├── id.ac.polban.model/

│ └── Produk.java

├── id.ac.polban.service/

│ ├── KoperasiManager.java

│ └── Transaksi.java

└── KoperasiApp.java (default package)

**Keuntungan Penggunaan Package:**

* **Organisasi Kode**: Memisahkan layer model dan service
* **Namespace Management**: Menghindari konflik nama kelas
* **Akses Control**: Mengatur visibilitas kelas dan member
* **Maintainability**: Memudahkan maintenance dan pengembangan

**1.2 Implementasi Static**

**1.2.1 Static Variables**

**Class Produk:**

private static int totalProduk = 0;

* **Fungsi**: Counter untuk menghitung total produk yang dibuat
* **Shared**: Semua instance Produk berbagi variable ini
* **Lifecycle**: Hidup selama aplikasi berjalan

**Class Transaksi:**

java

private static int nomorTransaksi = 1;

* **Fungsi**: Auto-increment counter untuk ID transaksi
* **Unique**: Memastikan setiap transaksi memiliki nomor unik

**Class KoperasiManager:**

private static KoperasiManager instance = null;

* **Fungsi**: Implementasi Singleton pattern
* **Purpose**: Memastikan hanya ada satu instance KoperasiManager

**1.2.2 Static Methods**

**Class Produk:**

public static int GetTotalProduk() {

return totalProduk;

}

* **Utility Method**: Dapat dipanggil tanpa membuat instance
* **Access**: Memberikan akses ke static variable totalProduk

**Class Transaksi:**

public static String buatNomorTransaksi() {

String nomor = "TRX -" + String.format("%03d", nomorTransaksi);

nomorTransaksi++;

return nomor;

}

* **Factory Method**: Membuat ID transaksi yang unik
* **Auto-increment**: Mengelola penomoran otomatis

**Class KoperasiManager:**

public static KoperasiManager getInstance() {

if (instance == null) {

instance = new KoperasiManager();

}

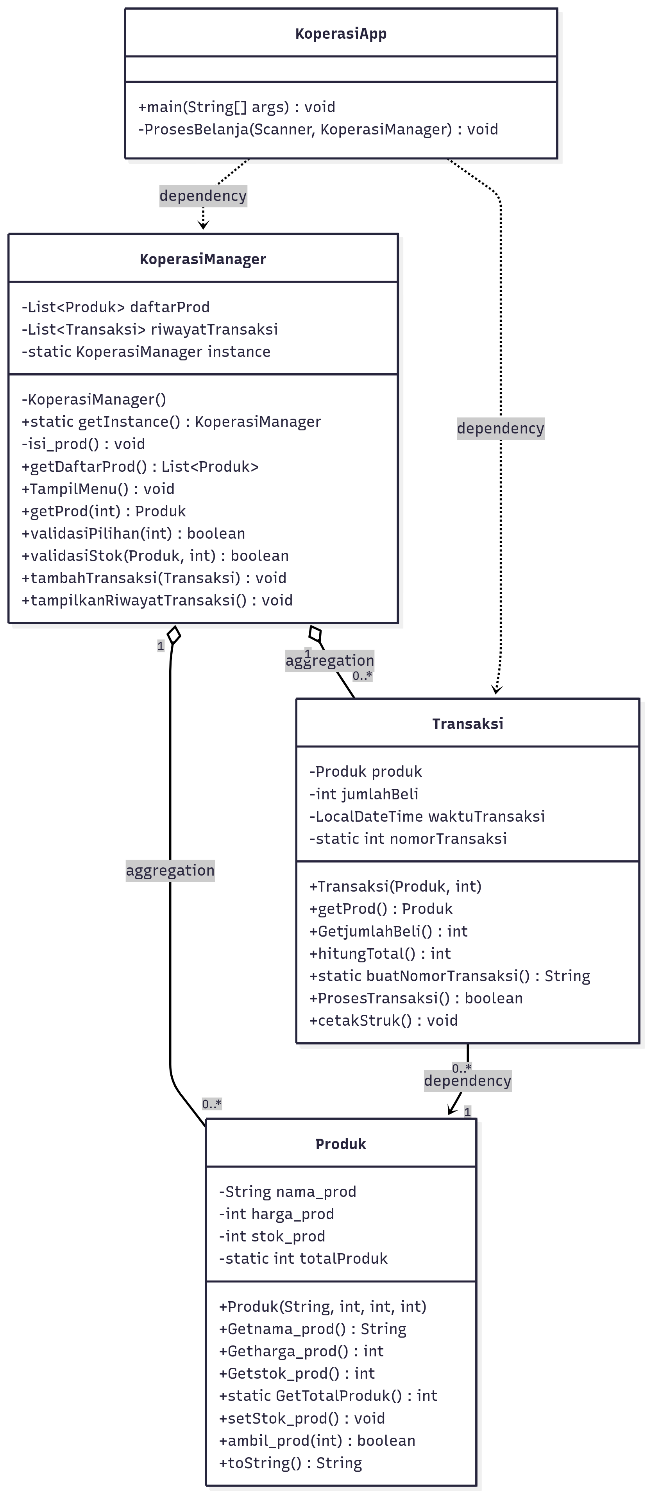
return instance;

}

* **Singleton Access**: Satu-satunya cara untuk mendapatkan instance
* **Lazy Initialization**: Instance dibuat saat pertama kali dibutuhkan

**2. CLASS DIAGRAM**

**2.1 Gambar Class Diagram**



**3. IMPLEMENTASI RELASI ANTAR KELAS**

**3.1 Dependency (Uses Relationship)**

**3.1.1 KoperasiApp → KoperasiManager**

public class KoperasiApp {

public static void main(String[] args) {

KoperasiManager koperasi = KoperasiManager.getInstance();

*// KoperasiApp menggunakan KoperasiManager*

}

}

**Karakteristik:**

* **Temporary**: Relasi bersifat sementara dalam method
* **Uses**: KoperasiApp menggunakan services dari KoperasiManager
* **No Ownership**: KoperasiApp tidak memiliki KoperasiManager

**3.1.2 KoperasiApp → Transaksi**

private static void ProsesBelanja(Scanner input, KoperasiManager koperasi) {

Transaksi transaksi = new Transaksi(produkdipilih, jumlah);

*// KoperasiApp membuat dan menggunakan Transaksi*

}

**Karakteristik:**

* **Creates**: KoperasiApp membuat instance Transaksi
* **Short-lived**: Object Transaksi digunakan dalam scope method
* **Functional**: Untuk melakukan proses bisnis tertentu

**3.1.3 Transaksi → Produk**

public class Transaksi {

public boolean ProsesTransaksi() {

return produk.ambil\_prod(jumlahBeli);

*// Transaksi menggunakan method dari Produk*

}

}

**Karakteristik:**

* **Method Invocation**: Transaksi memanggil method Produk
* **Behavioral**: Untuk melakukan operasi bisnis
* **Interface Usage**: Menggunakan public interface Produk

**3.2 Aggregation (Has-A Relationship)**

**3.2.1 KoperasiManager ○ Produk (Has-Many)**

public class KoperasiManager {

private List<Produk> daftarProd;

*// KoperasiManager HAS-A collection of Produk*

private void isi\_prod() {

daftarProd.add(new Produk("Pulpen", 3500, 10, 0));

*// KoperasiManager mengelola lifecycle Produk*

}

}

**Karakteristik:**

* **Whole-Part**: KoperasiManager adalah whole, Produk adalah part
* **Lifecycle Control**: KoperasiManager mengelola creation Produk
* **Independent Existence**: Produk bisa exist tanpa KoperasiManager
* **Collection Management**: Menggunakan List untuk menyimpan multiple Produk

**3.2.2 KoperasiManager ○ Transaksi (Has-Many)**

public class KoperasiManager {

private List<Transaksi> riwayatTransaksi;

*// KoperasiManager HAS-A collection of Transaksi*

public void tambahTransaksi(Transaksi transaksi) {

riwayatTransaksi.add(transaksi);

*// Menyimpan Transaksi untuk riwayat*

}

}

**Karakteristik:**

* **Storage Responsibility**: KoperasiManager menyimpan history Transaksi
* **Weak Ownership**: Transaksi bisa exist independent
* **Historical Data**: Untuk keperluan reporting dan audit trail

**3.3 Penjelasan Detail Relasi**

**3.3.1 Dependency vs Aggregation**

| **Aspek** | **Dependency** | **Aggregation** |
| --- | --- | --- |
| **Ownership** | Tidak ada kepemilikan | Ada kepemilikan sementara |
| **Lifecycle** | Independent | Partly dependent |
| **Relationship** | Uses/Calls | Has-A |
| **Duration** | Method scope | Object lifetime |
| **Multiplicity** | Usually 1:1 | Often 1:many |

**3.3.2 Mengapa Tidak Composition?**

Sistem ini menggunakan **Aggregation** bukan **Composition** karena:

* Produk dan Transaksi bisa exist secara independent
* Tidak ada strong ownership relationship
* Parts tidak di-destroy ketika whole di-destroy

**4. SOURCE CODE LENGKAP**

Link Github : https://github.com/dzhax499/Object-Oriented-SMT3/tree/main/W3-PRAKTIKUM

**4.1 Package id.ac.polban.model**

**4.1.1 Produk.java**

package id.ac.polban.model;

public class Produk {

private String nama\_prod;

private int harga\_prod;

private int stok\_prod;

*// Static variable untuk menghitung total produk*

private static int totalProduk = 0;

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;

totalProduk++; *// Increment setiap kali produk dibuat*

}

*// GETTER METHODS*

public String Getnama\_prod() {

return nama\_prod;

}

public int Getharga\_prod() {

return harga\_prod;

}

public int Getstok\_prod() {

return stok\_prod;

}

*// STATIC METHOD - dapat dipanggil tanpa instance*

public static int GetTotalProduk() {

return totalProduk;

}

*// SETTER METHODS*

public void setStok\_prod(int stok\_prod) {

this.stok\_prod = stok\_prod;

}

*// BUSINESS METHOD*

public boolean ambil\_prod(int jumlah) {

if (stok\_prod >= jumlah) {

stok\_prod = stok\_prod - jumlah;

return true;

}

return false;

}

@Override

public String toString() {

return nama\_prod + " - Rp" + harga\_prod + " (Stok: " + stok\_prod + ")";

}

}

**4.2 Package id.ac.polban.service**

**4.2.1 KoperasiManager.java**

package id.ac.polban.service;

import id.ac.polban.model.Produk;

import java.util.ArrayList;

import java.util.List;

public class KoperasiManager {

*// AGGREGATION: KoperasiManager HAS-A List of Produk*

private List<Produk> daftarProd;

*// AGGREGATION: KoperasiManager HAS-A List of Transaksi*

private List<Transaksi> riwayatTransaksi;

*// SINGLETON PATTERN - Static variable*

private static KoperasiManager instance = null;

*// SINGLETON PATTERN - Static method*

public static KoperasiManager getInstance() {

if (instance == null) {

instance = new KoperasiManager();

}

return instance;

}

*// Private constructor untuk Singleton*

private KoperasiManager() {

daftarProd = new ArrayList<>();

riwayatTransaksi = new ArrayList<>();

isi\_prod(); *// Initialize produk*

}

*// Method untuk inisialisasi produk*

private void isi\_prod() {

daftarProd.add(new Produk("Pulpen", 3500, 10, 0));

daftarProd.add(new Produk("Buku", 7000, 15, 0));

daftarProd.add(new Produk("Pensil", 3000, 7, 0));

daftarProd.add(new Produk("Penghapus", 2000, 20, 0));

}

public List<Produk> getDaftarProd() {

return daftarProd;

}

public void TampilMenu() {

System.out.println("========== MENU PRODUK KOPERASI ==========");

for (int i = 0; i < daftarProd.size(); i++) {

System.out.println((i + 1) + ". " + daftarProd.get(i).toString());

}

System.out.println("==========================================");

*// Menggunakan static method dari Produk*

System.out.println("Total jenis produk: " + Produk.GetTotalProduk());

System.out.println("==========================================");

}

public Produk getProd(int index) {

if (index >= 1 && index <= daftarProd.size()) {

return daftarProd.get(index - 1);

}

return null;

}

public boolean validasiPilihan(int pilihan) {

return pilihan >= 1 && pilihan <= daftarProd.size();

}

public boolean validasiStok(Produk produk, int jumlah) {

return produk.Getstok\_prod() >= jumlah;

}

*// AGGREGATION: Menambahkan Transaksi ke collection*

public void tambahTransaksi(Transaksi transaksi) {

riwayatTransaksi.add(transaksi);

}

public void tampilkanRiwayatTransaksi() {

System.out.println("========== RIWAYAT TRANSAKSI ==========");

if (riwayatTransaksi.isEmpty()) {

System.out.println("Belum ada transaksi.");

} else {

for (Transaksi trx : riwayatTransaksi) {

System.out.println("Produk: " + trx.getProd().Getnama\_prod() +

" | Qty: " + trx.GetjumlahBeli() +

" | Total: Rp." + trx.hitungTotal());

}

}

System.out.println("=======================================");

}

}

**4.2.2 Transaksi.java**

package id.ac.polban.service;

import id.ac.polban.model.Produk;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class Transaksi {

*// DEPENDENCY: Transaksi menggunakan Produk*

private Produk produk;

private int jumlahBeli;

private LocalDateTime waktuTransaksi;

*// Static variable untuk auto-increment nomor transaksi*

private static int nomorTransaksi = 1;

public Transaksi(Produk produk, int jumlahBeli) {

this.produk = produk;

this.jumlahBeli = jumlahBeli;

this.waktuTransaksi = LocalDateTime.now();

}

public Produk getProd() {

return produk;

}

public int GetjumlahBeli() {

return jumlahBeli;

}

public int hitungTotal() {

return produk.Getharga\_prod() \* jumlahBeli;

}

*// STATIC METHOD untuk generate nomor transaksi*

public static String buatNomorTransaksi() {

String nomor = "TRX-" + String.format("%03d", nomorTransaksi);

nomorTransaksi++;

return nomor;

}

*// DEPENDENCY: Menggunakan method dari Produk*

public boolean ProsesTransaksi() {

return produk.ambil\_prod(jumlahBeli);

}

public void cetakStruk() {

System.out.println("========== STRUK KOPERASI ==========");

System.out.println("No. Transaksi : " + buatNomorTransaksi());

System.out.println("Waktu : " +

waktuTransaksi.format(DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss")));

System.out.println("===================================");

System.out.println("Nama Barang : " + produk.Getnama\_prod());

System.out.println("Harga Satuan : Rp. " + produk.Getharga\_prod());

System.out.println("Jumlah Beli : " + jumlahBeli);

System.out.println("-----------------------------------");

System.out.println("Total Bayar : Rp. " + hitungTotal());

System.out.println("===================================");

System.out.println("Sisa Stok : " + produk.Getstok\_prod());

System.out.println("===================================");

}

}

**4.3 Main Application**

**4.3.1 KoperasiApp.java**

import id.ac.polban.model.Produk;

import id.ac.polban.service.KoperasiManager;

import id.ac.polban.service.Transaksi;

import java.util.Scanner;

public class KoperasiApp {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

*// DEPENDENCY: main method menggunakan KoperasiManager*

KoperasiManager koperasi = KoperasiManager.getInstance();

while (true) {

System.out.println("\n========== APLIKASI KOPERASI ==========");

System.out.println("1. Belanja");

System.out.println("2. Lihat Riwayat Transaksi");

System.out.println("3. Keluar");

System.out.print("Pilih menu: ");

int menu = input.nextInt();

switch (menu) {

case 1:

ProsesBelanja(input, koperasi);

break;

case 2:

koperasi.tampilkanRiwayatTransaksi();

break;

case 3:

System.out.println("Sampai jumpa... :)");

input.close();

return;

default:

System.out.println("Pilihan tidak valid!");

break;

}

}

}

private static void ProsesBelanja(Scanner input, KoperasiManager koperasi) {

koperasi.TampilMenu();

System.out.print("Pilih barang : ");

int pilihan = input.nextInt();

*// Validasi pilihan*

if (!koperasi.validasiPilihan(pilihan)) {

System.out.println("Pilihan tidak tersedia");

return;

}

*// DEPENDENCY: Mendapatkan Produk dari KoperasiManager*

Produk produkdipilih = koperasi.getProd(pilihan);

System.out.print("Jumlah Barang : ");

int jumlah = input.nextInt();

if (jumlah <= 0) {

System.out.println("Jumlah tidak boleh kurang dari 1.... :)");

return;

}

*// Validasi stok*

if (produkdipilih.Getstok\_prod() < jumlah) {

System.out.println("Jumlah stok tidak mencukupi.... :) " +

"Stok tersedia: " + produkdipilih.Getstok\_prod());

return;

}

*// DEPENDENCY: Membuat instance Transaksi*

Transaksi transaksi = new Transaksi(produkdipilih, jumlah);

*// Proses transaksi*

if (transaksi.ProsesTransaksi()) {

transaksi.cetakStruk();

*// AGGREGATION: KoperasiManager menyimpan Transaksi*

koperasi.tambahTransaksi(transaksi);

System.out.println("TRANSAKSI BERHASIL... :)");

} else {

System.out.println("TRANSAKSI GAGAL... :(");

}

}

}

**5. GENERATE JAR FILE**

**5.1 Struktur Project**

koperasi-project/

├── src/

│ ├── id/ac/polban/model/

│ │ └── Produk.java

│ ├── id/ac/polban/service/

│ │ ├── KoperasiManager.java

│ │ └── Transaksi.java

│ └── KoperasiApp.java

├── build/

└── koperasi.jar

**5.2 Langkah-langkah Generate JAR**

**5.2.1 Compile Source Code**

*javac -d build src/id/ac/polban/model/\*.java; if ($?) { javac -d build -cp build src/id/ac/polban/service/\*.java; if ($?) { javac -d build -cp build src/KoperasiApp.java; if ($?) { cd build; jar cvfe koperasi.jar KoperasiApp .; cd .. } } }*

**5.2.2 Create Manifest File**

Manifest-Version: 1.0

Main-Class: KoperasiApp

**5.2.3 Create JAR File**

*jar cfm koperasi.jar manifest.txt -C build .*

**5.2.4 Run JAR File**

java -jar koperasi.jar

**5.3 Project dengan JAR File Only**

**5.3.1 Struktur New Project**

koperasi-standalone/

├── koperasi.jar

├── README.md

└── run.bat (untuk Windows)

**5.3.2 run.bat**

@echo off

echo Starting Koperasi Application...

java -jar koperasi.jar

pause

**5.3.3 README.md**

# Aplikasi Koperasi

## Requirements

- Java 8 atau versi lebih tinggi

## Menjalankan Aplikasi

### Windows:

Double-click `run.bat` atau jalankan:

java -jar koperasi.jar

### Linux/Mac:

java -jar koperasi.jar

## Features

1. Belanja produk koperasi

2. Melihat riwayat transaksi

3. Manajemen stok otomatis

**6. KESIMPULAN**

**6.1 Implementasi Konsep OOP**

**6.1.1 Static Implementation**

* **Static Variables**: Berhasil diimplementasikan untuk counter dan singleton
* **Static Methods**: Digunakan untuk utility functions dan factory methods
* **Memory Efficiency**: Shared data across instances

**6.1.2 Package Organization**

* **Separation of Concerns**: Model dan Service terpisah dengan baik
* **Modularity**: Kode terorganisir secara logis
* **Reusability**: Package dapat di-reuse di project lain

**6.1.3 Class Relationships**

* **Dependency**: Clean interface usage tanpa tight coupling
* **Aggregation**: Proper whole-part relationship dengan lifecycle management
* **Design Patterns**: Singleton pattern terimplementasi dengan benar

**6.2 Kelebihan Implementasi**

* Struktur kode yang clean dan maintainable
* Proper error handling dan validation
* User-friendly interface
* Scalable architecture

**6.3 Pengembangan Lanjutan Nantinya**

* Implementasi database untuk persistent storage
* Unit testing untuk reliability
* Configuration file untuk flexibility
* Logging mechanism untuk debugging