**LAPORAN PRAKTIKUM**

**Inheritance,Override,Super**



**Disusun Oleh:**

**Dzakir Tsabit Asy Syafiq (241511071)**

**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika**

**Politeknik Negeri Bandung**

**23/09/2025**

**Kode modifikasi :**

**File: Produk.java**

package id.ac.polban.model;

/\*\*

\* Kelas Produk merepresentasikan item-item yang dijual di koperasi.

\* Ini adalah kelas dasar (superclass) yang menyimpan informasi umum produk.

\*/

public class Produk {

private String nama\_prod;

private int harga\_prod;

private int stok\_prod;

// static

private static int totalProduk = 0;

/\*\*

\* Konstruktor untuk membuat objek Produk baru.

\* @param nama\_prod Nama produk.

\* @param harga\_prod Harga produk per unit.

\* @param stok\_prod Jumlah stok produk yang tersedia.

\*/

public Produk(String nama\_prod, int harga\_prod, int stok\_prod) {

this.nama\_prod = nama\_prod;

this.harga\_prod = harga\_prod;

this.stok\_prod = stok\_prod;

totalProduk++;

}

// GETTER

/\*\*

\* Mengembalikan nama produk.

\* @return Nama produk.

\*/

public String Getnama\_prod() {

return nama\_prod;

}

/\*\*

\* Mengembalikan harga produk.

\* Ini adalah metode yang akan di-override di subclass.

\* @return Harga produk per unit.

\*/

public int Getharga\_prod() {

return harga\_prod;

}

/\*\*

\* Mengembalikan jumlah stok produk yang tersedia.

\* @return Jumlah stok produk.

\*/

public int Getstok\_prod() {

return stok\_prod;

}

// STATIC METHOD

/\*\*

\* Mengembalikan total jumlah produk yang telah dibuat.

\* @return Total produk.

\*/

public static int GetTotalProduk() {

return totalProduk;

}

// SETTER

/\*\*

\* Mengatur jumlah stok produk.

\* @param stok\_prod Jumlah stok baru.

\*/

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

this.stok\_prod = stok\_prod;

}

// METHOD

/\*\*

\* Mengurangi stok produk setelah transaksi berhasil.

\* @param jumlah Jumlah produk yang dibeli.

\* @return True jika stok mencukupi dan berhasil dikurangi, false sebaliknya.

\*/

public boolean kurangi\_prod(int jumlah){

if (stok\_prod >= jumlah){

stok\_prod = stok\_prod - jumlah;

return true;

}

return false;

}

/\*\*

\* Mengembalikan representasi string dari objek Produk.

\* @return String yang berisi nama, harga, dan stok produk.

\*/

@Override

public String toString() {

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

}

}

**File: ProdukDiskon.java**

package id.ac.polban.model;

/\*\*

\* Kelas ProdukDiskon adalah subclass dari Produk yang menambahkan fungsionalitas diskon.

\* Kelas ini mewarisi semua atribut dan metode dari Produk.

\*/

public class ProdukDiskon extends Produk {

private double persenDiskon;

/\*\*

\* Konstruktor untuk membuat objek ProdukDiskon.

\* Memanggil konstruktor superclass Produk untuk inisialisasi properti dasar.

\* @param nama\_prod Nama produk.

\* @param harga\_prod Harga produk per unit.

\* @param stok\_prod Jumlah stok produk.

\* @param persenDiskon Persentase diskon yang diberikan (dalam persen, misal 10.0).

\*/

public ProdukDiskon(String nama\_prod, int harga\_prod, int stok\_prod, double persenDiskon) {

// Memanggil konstruktor superclass dengan keyword 'super'

super(nama\_prod, harga\_prod, stok\_prod);

this.persenDiskon = persenDiskon;

}

/\*\*

\* Meng-override metode Getharga\_prod() dari superclass Produk.

\* Metode ini menghitung harga produk setelah dikurangi diskon.

\* @return Harga produk setelah diskon.

\*/

@Override

public int Getharga\_prod() {

// Menggunakan 'super' untuk mendapatkan harga dasar dari superclass

double hargaAsli = super.Getharga\_prod();

double hargaSetelahDiskon = hargaAsli - (hargaAsli \* persenDiskon / 100);

return (int) Math.round(hargaSetelahDiskon);

}

/\*\*

\* Mengembalikan representasi string dari objek ProdukDiskon, termasuk informasi diskon.

\* @return String yang berisi nama, harga, stok, dan diskon produk.

\*/

@Override

public String toString() {

return super.Getnama\_prod() + " - Rp" + Getharga\_prod() + " (Diskon " + persenDiskon + "%)" + " (Stok: " + super.Getstok\_prod() + ")";

}

}

**File: KoperasiManager.java**

package id.ac.polban.service;

import id.ac.polban.model.Produk;

import id.ac.polban.model.ProdukDiskon; // Import kelas ProdukDiskon

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Kelas KoperasiManager mengelola data produk dan riwayat transaksi koperasi.

\* Mengimplementasikan pola Singleton untuk memastikan hanya ada satu instance.

\*/

public class KoperasiManager {

private List<Produk> daftarProd;

private List<Transaksi> riwayatTransaksi;

// STATIC VARIABLES (Singleton Pattern)

private static KoperasiManager instance;

// static method

/\*\*

\* Mengembalikan instance tunggal dari KoperasiManager (Singleton).

\* @return Instance KoperasiManager.

\*/

public static KoperasiManager getInstance() {

if (instance == null ) {

instance = new KoperasiManager();

}

return instance;

}

/\*\*

\* Konstruktor privat untuk mencegah instansiasi dari luar.

\*/

private KoperasiManager(){

daftarProd = new ArrayList<>();

riwayatTransaksi = new ArrayList<>();

isi\_prod();

}

/\*\*

\* Mengisi daftar produk dengan beberapa item default.

\* Ditambahkan objek ProdukDiskon baru.

\*/

private void isi\_prod() {

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

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

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

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

// Tambahan: Contoh produk diskon untuk demonstrasi Inheritance

daftarProd.add(new ProdukDiskon("Pulpen Diskon", 3500, 5, 10.0));

}

/\*\*

\* Mengembalikan daftar produk yang tersedia.

\* @return List objek Produk.

\*/

public List<Produk> getDaftarProd(){

return daftarProd;

}

/\*\*

\* Menampilkan menu produk kepada pengguna.

\*/

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("==========================================");

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

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

}

/\*\*

\* Mendapatkan objek Produk berdasarkan pilihan indeks.

\* @param index Pilihan indeks dari menu.

\* @return Objek Produk jika valid, null jika tidak.

\*/

public Produk getProd(int index) {

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

return daftarProd.get(index -1 );

}

return null;

}

/\*\*

\* Melakukan validasi pilihan menu produk dari user.

\* @param pilihan Pilihan menu yang dimasukkan user.

\* @return True jika pilihan valid, false sebaliknya.

\*/

public boolean validasiPilihan(int pilihan) {

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

}

/\*\*

\* Melakukan validasi stok produk.

\* @param produk Objek produk yang dipilih.

\* @param jumlah Jumlah yang ingin dibeli.

\* @return True jika stok mencukupi, false sebaliknya.

\*/

public boolean validasiStok(Produk produk, int jumlah) {

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

}

/\*\*

\* Menambahkan transaksi ke dalam riwayat.

\* @param transaksi Objek transaksi yang akan ditambahkan.

\*/

public void tambahTransaksi(Transaksi transaksi){

riwayatTransaksi.add(transaksi);

}

/\*\*

\* Menampilkan riwayat transaksi yang telah dilakukan.

\*/

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("No: " + trx.getProd() + " | " +

trx.getProd().Getnama\_prod() + " | " +

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

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

}

}

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

}

}

**File: KoperasiApp.java**

import id.ac.polban.model.Produk;

import id.ac.polban.service.KoperasiManager;

import id.ac.polban.service.Transaksi;

import java.util.Scanner;

/\*\*

\* KoperasiApp adalah kelas utama untuk menjalankan aplikasi koperasi sederhana.

\* Mengelola interaksi user, proses belanja, dan menampilkan riwayat transaksi.

\*/

public class KoperasiApp {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Mendapatkan instance KoperasiManager (Singleton)

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... :)");

return;

default:

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

break;

}

}

}

/\*\*

\* Metode untuk memproses alur belanja produk.

\* @param input Objek Scanner untuk input user.

\* @param koperasi Objek KoperasiManager untuk mengelola data.

\*/

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

koperasi.TampilMenu();

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

int pilihan = input.nextInt();

// Error handling atau validasi pilihan

if (!koperasi.validasiPilihan(pilihan)) {

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

return;

}

// Ambil Barang yang dipilih

Produk produkdipilih = koperasi.getProd(pilihan);

if (produkdipilih == null) {

System.out.println("Produk tidak ditemukan, silakan pilih kembali.");

return;

}

// jumlah?

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

int jumlah = input.nextInt();

if (jumlah <= 0) {

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

return;

}

// cek stok

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

System.out.println("Jumlah stok tidak mencukup.... :) cuma ada stok : " + produkdipilih.Getstok\_prod());

return;

}

// buat transaksi

Transaksi transaksi = new Transaksi(produkdipilih , jumlah);

// proses pembelian

if (transaksi.ProsesTransaksi()){

transaksi.cetakStruk();

koperasi.tambahTransaksi(transaksi);

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

} else {

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

}

}

}

**File: Transaksi.java**

package id.ac.polban.service;

import id.ac.polban.model.Produk;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

/\*\*

\* Kelas Transaksi merepresentasikan satu kali transaksi pembelian produk.

\*/

public class Transaksi {

private Produk produk;

private int jumlahBeli;

private LocalDateTime waktuTransaksi;

// static variable

private static int nomorTransaksi = 1; // auto-increment nomor

/\*\*

\* Konstruktor untuk membuat objek Transaksi baru.

\* @param produk Objek produk yang dibeli.

\* @param jumlahBeli Jumlah unit yang dibeli.

\*/

public Transaksi(Produk produk,int jumlahBeli){

this.produk = produk;

this.jumlahBeli = jumlahBeli;

this.waktuTransaksi = LocalDateTime.now();

}

/\*\*

\* Mengembalikan objek produk yang dibeli dalam transaksi ini.

\* @return Objek Produk.

\*/

public Produk getProd(){

return produk;

}

/\*\*

\* Mengembalikan jumlah barang yang dibeli.

\* @return Jumlah unit produk.

\*/

public int GetjumlahBeli(){

return jumlahBeli;

}

/\*\*

\* Menghitung total harga transaksi.

\* @return Total harga.

\*/

public int hitungTotal() {

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

}

//static Method

/\*\*

\* Membuat nomor transaksi yang unik secara auto-increment.

\* @return String nomor transaksi.

\*/

public static String buatNomorTransaksi(){

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

nomorTransaksi++;

return nomor;

}

/\*\*

\* Memproses transaksi dengan mengurangi stok produk.

\* @return True jika proses berhasil, false sebaliknya.

\*/

public boolean ProsesTransaksi(){

return produk.kurangi\_prod(jumlahBeli);

}

/\*\*

\* Mencetak struk transaksi ke konsol.

\*/

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("===================================");

}

}

**Konsep yang Diterapkan**

**Inheritance (Pewarisan)**

Konsep ini memungkinkan sebuah kelas (ProdukDiskon) untuk mewarisi atribut dan metode dari kelas lain (Produk), sehingga menghindari duplikasi kode. Kelas ProdukDiskon akan menjadi **subclass** atau **child class**, sedangkan Produk akan menjadi **superclass** atau **parent class**.

**Method Overriding**

Ini adalah kemampuan subclass untuk menyediakan implementasi spesifik dari metode yang sudah ada di superclass-nya. Dalam kasus ini, meng-override metode Getharga\_prod() di kelas ProdukDiskon untuk menghitung harga setelah diskon.

**Keyword super**

Keyword ini digunakan di dalam subclass untuk merujuk pada anggota (variabel atau metode) dari superclass-nya. Kita akan menggunakan super di konstruktor ProdukDiskon untuk memanggil konstruktor dari kelas Produk dan di metode Getharga\_prod() yang di-override untuk mendapatkan harga asli sebelum diskon.

**Daftar Modifikasi**

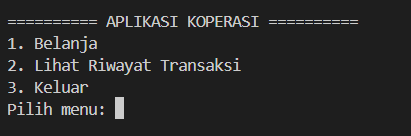
1. **Membuat Kelas ProdukDiskon**:
   * Sebuah kelas baru, ProdukDiskon.java, dibuat di dalam package id.ac.polban.model.
   * Kelas ini mewarisi kelas Produk menggunakan sintaks extends Produk.
   * Kelas ini memiliki atribut baru persenDiskon untuk menyimpan persentase diskon.
2. **Modifikasi Konstruktor ProdukDiskon**:
   * Konstruktor kelas ProdukDiskon akan menerima parameter tambahan untuk persentase diskon.
   * Di dalam konstruktor ini, super(nama\_prod, harga\_prod, stok\_prod) dipanggil. Ini memastikan bahwa konstruktor dari kelas Produk dijalankan terlebih dahulu untuk menginisialisasi atribut yang diwarisi (nama\_prod, harga\_prod, dan stok\_prod).
3. **Menerapkan Method Overriding**:
   * Metode Getharga\_prod() dari kelas Produk di-override di dalam kelas ProdukDiskon.
   * Metode yang di-override ini akan menghitung harga baru dengan menerapkan diskon. Ini menggunakan super.Getharga\_prod() untuk mendapatkan harga dasar dari superclass, lalu menguranginya dengan nilai diskon.
   * Anatomi @Override ditambahkan di atas metode untuk menandakan bahwa metode tersebut sengaja di-override, yang juga membantu compiler dalam mendeteksi kesalahan jika ada ketidaksesuaian.
4. **Modifikasi KoperasiManager.java**:
   * Untuk menunjukkan fungsionalitas ProdukDiskon, kita menambahkan objek ProdukDiskon baru ke daftarProd dalam metode isi\_prod(). Contohnya adalah "Pulpen Diskon" dengan diskon 10%.
5. **Menambahkan Komentar Dokumentasi (Javadoc)**:
   * Komentar Javadoc (/\*\* ... \*/) ditambahkan ke setiap kelas (KoperasiApp, KoperasiManager, Produk, ProdukDiskon, Transaksi) dan metode utama. Ini menjelaskan tujuan dari setiap kelas, atribut, metode, dan parameter, meningkatkan kejelasan kode dan kemudahannya untuk dipahami oleh pengembang lain.

**Hasil Implementasi**

Modifikasi ini memungkinkan sistem untuk menangani berbagai jenis produk secara lebih fleksibel. Sekarang, kita dapat memiliki produk biasa dan produk diskon, di mana perhitungan harga untuk produk diskon akan otomatis menggunakan logika yang di-override. Ini menunjukkan bagaimana inheritance dan overriding dapat digunakan untuk memperluas fungsionalitas tanpa memodifikasi kode kelas induk yang sudah ada, sesuai dengan prinsip **Open/Closed Principle** dari SOLID.

**OUTPUT:**

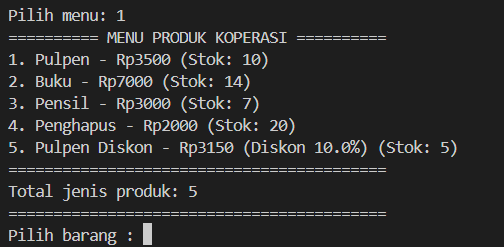
**1. Output: Tampilan Menu Utama**

****

**Penjelasan code**: Output ini dihasilkan oleh metode main di kelas **KoperasiApp.java**

* while (true): Loop ini memastikan menu utama akan terus muncul sampai pengguna memilih "Keluar".
* System.out.println digunakan untuk mencetak baris-baris teks statis yang membentuk menu.
* input.nextInt(): Baris ini menunggu input dari pengguna, yaitu 1 untuk "Belanja".

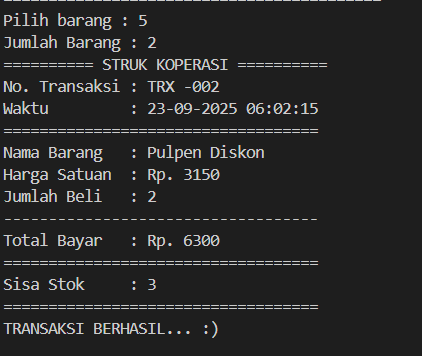
**2. Output: Tampilan Menu Produk**

****

**Penjelasan Kode**: Output ini adalah hasil dari pemanggilan koperasi.TampilMenu() setelah pengguna memilih menu 1.

* KoperasiManager membuat instance-nya (getInstance()) dan memanggil isi\_prod() di konstruktornya. Di sinilah **inheritance** dan **polymorphism** bekerja.
* daftarProd.add(new ProdukDiskon("Pulpen Diskon", 3500, 5, 10.0)) di isi\_prod() menambahkan objek **ProdukDiskon** ke dalam List<Produk>. Ini dimungkinkan karena ProdukDiskon adalah subclass dari Produk.
* koperasi.TampilMenu() memanggil daftarProd.get(i).toString() untuk setiap produk.
* Untuk produk biasa, toString() dari kelas **Produk** yang dipanggil.
* Untuk "Pulpen Diskon", toString() dari kelas **ProdukDiskon** yang di-override yang dipanggil. Metode ini mencetak harga diskon dan persentase diskon.
* **Harga diskon (Rp3150)** dihitung di dalam metode Getharga\_prod() yang di-override pada kelas ProdukDiskon: (3500 - (3500 \* 10 / 100)) = 3150.
* Produk.GetTotalProduk(): Metode statis ini mengembalikan total produk yang telah dibuat, yang bertambah setiap kali sebuah objek Produk atau ProdukDiskon (yang memanggil super() di konstruktornya) dibuat.

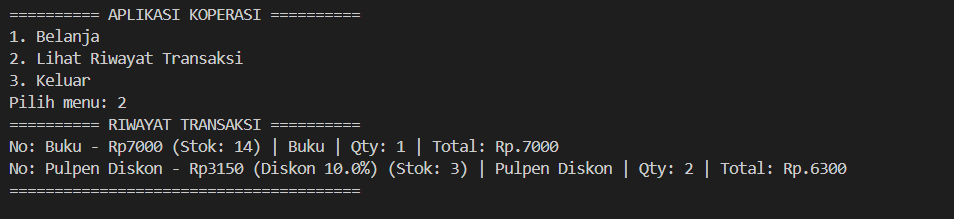
**3. Output: Proses Transaksi dan Struk**



**Penjelasan Kode**: Output ini dihasilkan setelah pengguna memilih barang (5) dan jumlah (2).

* Produk produkdipilih = koperasi.getProd(pilihan): Kode ini mengambil objek ProdukDiskon dari daftar produk.
* new Transaksi(produkdipilih, jumlah): Objek Transaksi dibuat, menyimpan referensi ke objek ProdukDiskon.
* transaksi.cetakStruk(): Metode ini mencetak detail transaksi.
* produk.Getharga\_prod(): Saat cetakStruk memanggil produk.Getharga\_prod() (melalui transaksi), **polymorphism** bekerja lagi. JVM secara otomatis memanggil metode Getharga\_prod() yang di-override dari kelas ProdukDiskon, bukan dari kelas Produk.
* hitungTotal(): Metode ini menghitung total harga: harga\_prod \* jumlahBeli yang menjadi 3150 \* 2 = 6300.
* transaksi.ProsesTransaksi(): Metode ini memanggil produk.kurangi\_prod(jumlah). Stok Pulpen Diskon berkurang dari 5 menjadi 3.
* koperasi.tambahTransaksi(transaksi): Objek Transaksi ditambahkan ke riwayatTransaksi.

**4. Output: Tampilan Riwayat Transaksi**

****

**Penjelasan Kode**: Output ini muncul setelah pengguna memilih menu 2.

* koperasi.tampilkanRiwayatTransaksi(): Metode ini mengiterasi riwayatTransaksi.
* trx.getProd().Getnama\_prod(): Baris ini mengambil nama produk dari objek transaksi, yang akan mengembalikan "Pulpen Diskon".
* trx.getProd(): Bagian ini akan mencetak representasi string dari objek ProdukDiskon.
* trx.hitungTotal(): Baris ini memanggil hitungTotal() yang sudah menyimpan hasil perhitungan harga diskon, sehingga total yang ditampilkan adalah Rp6300.