

Nama : Yunan Faila Sofi

NIM : H1D024033

ShiftLama : G

ShiftBaru : C

Membuat Interface Payment

```
/**
 * Interface ini mendefinisikan kontrak minimal untuk setiap metode pembayaran.
 */
public interface PaymentMethod {

    // Method untuk memproses transaksi pembayaran
    String processPayment();

    // Method untuk mengembalikan informasi detail transaksi
    String getPaymentDetails();

    // Method untuk mengembalikan biaya transaksi
    double getTransactionFee();

    // Method untuk mengembalikan saldo saat ini
    double getBalance();
}
```

Class EWalletPayment

```
package pert7;

/**
 * Class ini mengimplementasikan interface PaymentMethod untuk E-Wallet.
 */
public class EWalletPayment implements PaymentMethod {

    // =====
    // ATRIBUT/STATE
    // =====
    private String namaLayanan;
    private double nominalPembayaran;
    private double saldoAwal;
    private double saldoPengguna;
```

```

// Konstanta biaya transaksi (misal Rp 2.000)
private static final double FEE = 2000.0;

// =====
// CONSTRUCTOR
// =====

/**
 * Constructor untuk membuat objek EWalletPayment.
 * @param namaLayanan Nama layanan e-wallet (cth: OVO, Dana).
 * @param nominalPembayaran Jumlah yang harus dibayar.
 * @param saldoPengguna Saldo awal pengguna.
 */
public EWalletPayment(String namaLayanan, double nominalPembayaran, double
saldoPengguna) {
    this.namaLayanan = namaLayanan;
    this.nominalPembayaran = nominalPembayaran;
    this.saldoAwal = saldoPengguna; // Menyimpan saldo awal untuk ditampilkan
    this.saldoPengguna = saldoPengguna;
}

// =====
// IMPLEMENTASI METHOD DARI INTERFACE
// =====

@Override
public String processPayment() {
    // Total biaya: nominal pembayaran + biaya transaksi
    double totalBiaya = this.nominalPembayaran + getTransactionFee();

    // Pengecekan saldo
    if (this.saldoPengguna >= totalBiaya) {
        // Saldo cukup: kurangi saldo
        this.saldoPengguna -= totalBiaya;
        return "Pembayaran berhasil!";
    } else {
        // Saldo tidak cukup
        return "Pembayaran gagal! Saldo tidak cukup.";
    }
}

```

```

    }

    @Override
    public String getPaymentDetails() {
        return "Detail Transaksi: Pembayaran dilakukan melalui " + this.namaLayanan;
    }

    @Override
    public double getTransactionFee() {
        return FEE;
    }

    @Override
    public double getBalance() {
        return this.saldoPengguna;
    }

    // =====
    // GETTER UNTUK KEPENTINGAN DISPLAY
    // =====

    public double getSaldoAwal() {
        return this.saldoAwal;
    }

    public double getNominalPembayaran() {
        return this.nominalPembayaran;
    }
}

```

1. Alur Kerja Program (Workflow)

Program berjalan secara sekuensial dari kelas PaymentTest. Berikut adalah langkah-langkah logisnya:

- A. Inisialisasi Data: Program menetapkan variabel awal di method main: Saldo Rp 150.000, Tagihan Rp 50.000, dan Layanan "OVO".
- B. Instansiasi Objek (Penerapan Interface):
 - 1) Objek eWallet dibuat dari kelas EWalletPayment.
 - 2) Constructor dijalankan untuk menyimpan data awal.

C. Display Pra-Transaksi: Program menampilkan saldo awal dan nominal yang akan dibayar menggunakan *getter*.

D. Proses Pembayaran (*processPayment*):

1. Method ini dipanggil untuk melakukan eksekusi logika bisnis utama.
2. Langkah 1 (Hitung Biaya): Menjumlahkan nominalPembayaran (50.000) + FEE (2.000) = Total 52.000.
3. Langkah 2 (Cek Saldo): Program membandingkan apakah saldo pengguna (150.000) \geq total biaya (52.000).
4. Langkah 3 (Deduksi): Karena saldo cukup, saldo pengguna dikurangi 52.000 menjadi 98.000. String "Pembayaran berhasil!" dikembalikan.

E. Display Pasca-Transaksi:

1. Program menampilkan pesan hasil transaksi.
2. Program mengambil saldo terbaru via *getBalance()* (sekarang 98.000) dan menampilkannya.
3. Terakhir, program menampilkan detail penyedia layanan.

2. Fungsi dan Komponen yang Digunakan

Kode ini memperkenalkan konsep Interface, yang berbeda dengan *Abstract Class* pada pertemuan sebelumnya.

A. Interface *PaymentMethod* (Kontrak)

- Definisi: Interface hanya berisi deklarasi method tanpa body (isi). Ini bertindak sebagai "kontrak" atau aturan main.
- Fungsi: Menjamin bahwa kelas apapun yang mengimplementasikan interface ini (baik itu E-Wallet, Kartu Kredit, atau Transfer Bank) wajib memiliki 4 method utama: *processPayment*, *getPaymentDetails*, *getTransactionFee*, dan *getBalance*.

B. Kelas *EWalletPayment* (Implementasi)

- implements *PaymentMethod*: Keyword ini menandakan bahwa kelas ini menyetujui kontrak dari interface. Jika salah satu method interface tidak dibuat isinya di sini, program akan *error*.
- static final: Digunakan pada variabel FEE.
 - static: Milik kelas, bukan objek (hemat memori).
 - final: Nilainya tetap (konstanta) dan tidak bisa diubah setelah program berjalan.
- Logika Transaksi: Menggunakan if-else untuk memastikan saldo tidak menjadi negatif.

C. Kelas PaymentTest (Driver)

System.out.printf: Digunakan untuk mencetak angka dengan format yang rapi (tanpa desimal berlebih, menggunakan %.0f).

3. Hasil Output Program

```
=== PROGRAM SISTEM PEMBAYARAN (E-WALLET) ===  
Saldo awal: 150000  
Memproses pembayaran sebesar 50000...  
(Biaya Transaksi: 2000)  
  
Pembayaran berhasil!  
Sisa Saldo: 98000  
Detail Transaksi: Pembayaran dilakukan melalui OVO  
=====
```

Process finished with exit code 0

Analisis Output:

Perhatikan bahwa saldo berkurang bukan hanya sebesar 50.000, melainkan 52.000 karena adanya biaya admin (FEE) yang bersifat *hardcoded* (tetap) di dalam kelas EWalletPayment.