Muhammad Aqil Karomy
H1D024096
Shift KRS C
Shift Baru G

Pertemuan 7

DESKRIPSI TUGAS:
Membuat sistem pembayaran e-wallet menggunakan interface PaymentMethod yang berisi kontrak method untuk pemrosesan pembayaran. Interface akan diimplementasikan oleh class EWalletPayment yang menangani transaksi e-wallet dengan validasi saldo. Program ini mendemonstrasikan konsep interface, implementasi multiple method, dan polymorphism dalam Java.

KODE PROGRAM:

1. Interface PaymentMethod.java

```
public interface PaymentMethod {
   void processPayment();
   String getPaymentDetails();
   double getTransactionFee();
   double getBalance();
}
```

2. Class EWalletPayment.java

```
public class EWalletPayment implements PaymentMethod {
   private String providerName;
   private double saldo;
   private double nominalTransaksi;

   public EWalletPayment(String providerName, double saldo, double nominalTransaksi) {
      this.providerName = providerName;
      this.saldo = saldo;
      this.nominalTransaksi = nominalTransaksi;
   }

   public String getProviderName() {
      return providerName;
   }
```

```
    public void setProviderName(String providerName) {
        this.providerName = providerName;
    }

    public double getSaldo() {
        return saldo;
    }

    public void setSaldo(double saldo) {
        this.saldo = saldo;
    }

    public double getNominalTransaksi() {
        return nominalTransaksi;
    }

    public void setNominalTransaksi(double nominalTransaksi) {
        this.nominalTransaksi = nominalTransaksi;
    }

    @Override
    public void processPayment() {
        double totalBiaya = nominalTransaksi + getTransactionFee();
        if (saldo >= totalBiaya) {
            saldo -= totalBiaya;
            System.out.println("Pembayaran berhasil!");
            System.out.println("Total yang dibayar: Rp " + totalBiaya);
        } else {
            System.out.println("Pembayaran gagal! Saldo tidak mencukupi.");
            System.out.println("Saldo saat ini: Rp " + saldo);
            System.out.println("Total yang dibutuhkan: Rp " + totalBiaya);
        }
    }

    @Override
    public String getPaymentDetails() {
        return "Provider: " + providerName + " | Nominal Transaksi: Rp " + nominalTransaksi + "
| Biaya Admin: Rp " + getTransactionFee();
    }

    @Override
```

```
    public double getTransactionFee() {
        return nominalTransaksi * 0.01;
    }

    @Override
    public double getBalance() {
        return saldo;
    }
}
```

3. Main Class PaymentTest.java

```
public class PaymentTest {
    public static void main(String[] args) {
        System.out.println("=== SISTEM PEMBAYARAN E-WALLET ===");
        System.out.println();

        EWalletPayment gopay = new EWalletPayment("GoPay", 500000, 150000);

        System.out.println("--- DATA SEBELUM TRANSAKSI ---");
        System.out.println(gopay.getPaymentDetails());
        System.out.println("Saldo: Rp " + gopay.getBalance());
        System.out.println();

        System.out.println("--- PROSES PEMBAYARAN ---");
        gopay.processPayment();
        System.out.println();

        System.out.println("--- DATA SETELAH TRANSAKSI ---");
        System.out.println(gopay.getPaymentDetails());
        System.out.println("Saldo: Rp " + gopay.getBalance());
        System.out.println();

        EWalletPayment dana = new EWalletPayment("Dana", 50000, 100000);

        System.out.println("--- DATA SEBELUM TRANSAKSI (DANA) ---");
        System.out.println(dana.getPaymentDetails());
        System.out.println("Saldo: Rp " + dana.getBalance());
        System.out.println();

        System.out.println("--- PROSES PEMBAYARAN (DANA) ---");
```

```
        dana.processPayment();
    }
}
```

OUTPUT PROGRAM:

```
=== SISTEM PEMBAYARAN E-WALLET ===

--- DATA SEBELUM TRANSAKSI ---
Provider: GoPay | Nominal Transaksi: Rp 150000.0 | Biaya Admin: Rp 1500.0
Saldo: Rp 500000.0

--- PROSES PEMBAYARAN ---
Pembayaran berhasil!
Total yang dibayar: Rp 151500.0

--- DATA SETELAH TRANSAKSI ---
Provider: GoPay | Nominal Transaksi: Rp 150000.0 | Biaya Admin: Rp 1500.0
Saldo: Rp 348500.0

--- DATA SEBELUM TRANSAKSI (DANA) ---
Provider: Dana | Nominal Transaksi: Rp 100000.0 | Biaya Admin: Rp 1000.0
Saldo: Rp 50000.0

--- PROSES PEMBAYARAN (DANA) ---
Pembayaran gagal! Saldo tidak mencukupi.
Saldo saat ini: Rp 50000.0
Total yang dibutuhkan: Rp 101000.0
```

PENJELASAN ALUR PROGRAM:

1. Deklarasi Interface
   Interface PaymentMethod dideklarasikan dengan 4 method abstract tanpa implementasi. Interface ini berfungsi sebagai kontrak yang harus dipenuhi oleh semua class yang mengimplementasikannya.

2. Implementasi Class EWalletPayment
   Class EWalletPayment mengimplementasikan interface PaymentMethod dengan keyword "implements". Class ini wajib menyediakan implementasi untuk semua method yang dideklarasikan di interface, yaitu processPayment(), getPaymentDetails(), getTransactionFee(), dan getBalance().

3. Constructor dan Atribut

Constructor menerima 3 parameter (providerName, saldo, nominalTransaksi) untuk menginisialisasi objek e-wallet. Semua atribut bersifat private untuk encapsulation.

4. Method processPayment()

Method ini menghitung total biaya (nominal + fee 1%), kemudian mengecek apakah saldo mencukupi. Jika cukup, saldo dikurangi dan menampilkan pesan sukses. Jika tidak, menampilkan pesan gagal dengan detail saldo dan kebutuhan.

5. Method getPaymentDetails()

Method ini mengembalikan string yang berisi informasi lengkap transaksi: nama provider, nominal transaksi, dan biaya admin.

6. Method getTransactionFee()

Method ini menghitung biaya admin sebesar 1% dari nominal transaksi.

7. Method getBalance()

Method ini mengembalikan saldo terkini dari e-wallet.

8. Testing di Main

Program membuat 2 objek: gopay dengan saldo cukup (500rb untuk transaksi 150rb) dan dana dengan saldo tidak cukup (50rb untuk transaksi 100rb). Ini mendemonstrasikan kedua skenario success dan failure.

PENJELASAN FUNGSI METHOD:

1. processPayment()

Fungsi: Memproses pembayaran dengan validasi saldo. Menghitung total biaya termasuk fee, mengecek kecukupan saldo, dan melakukan debit jika valid.

2. getPaymentDetails()

Fungsi: Menyediakan informasi detail transaksi dalam format string yang readable. Berguna untuk logging atau display.

3. getTransactionFee()

Fungsi: Menghitung biaya admin sebagai persentase dari nominal. Dapat dengan mudah dimodifikasi jika aturan fee berubah.

4. getBalance()

Fungsi: Getter untuk mendapatkan saldo terkini. Penting untuk checking sebelum atau sesudah transaksi.

KONSEP OOP YANG DITERAPKAN:

1. Interface

Interface adalah kontrak yang mendefinisikan method signature tanpa implementasi. Dalam program ini, PaymentMethod menjadi blueprint yang harus diikuti oleh semua payment method. Interface tidak bisa diinstansiasi langsung dan semua method-nya bersifat abstract secara default.

2. Implementation

Keyword "implements" mendeklarasikan bahwa class akan mengimplementasikan interface tertentu. Class yang implements interface wajib menyediakan implementasi konkret untuk semua abstract method di interface tersebut. Jika tidak, akan terjadi compile error kecuali class dideklarasikan sebagai abstract.

3. Polymorphism

Interface memungkinkan polymorphism karena kita bisa mendeklarasikan variabel dengan tipe interface dan assign objek dari class yang mengimplementasikannya. Contoh: PaymentMethod payment = new EWalletPayment("GoPay", 100000, 50000). Ini memungkinkan fleksibilitas tinggi dalam design program.

4. Encapsulation

Semua atribut (providerName, saldo, nominalTransaksi) dibuat private dan diakses melalui getter/setter. Ini melindungi data dari akses langsung dan memungkinkan validasi di dalam setter jika diperlukan.

5. Abstraction

Interface menyediakan abstraction layer dengan menyembunyikan detail implementasi. Client code hanya perlu tahu method apa yang tersedia (processPayment, getBalance, dll) tanpa perlu tahu bagaimana implementasi internal-nya.

6. Multiple Implementation

Keuntungan interface adalah satu class bisa implements multiple interface (multiple inheritance). Ini berbeda dengan class inheritance yang hanya single inheritance. Contoh: class bisa implements PaymentMethod, Serializable, Comparable sekaligus.

7. @Override Annotation

Annotation @Override digunakan untuk menandai bahwa method tersebut meng-override method dari interface. Ini membantu compiler mendeteksi error jika signature method tidak sesuai dengan yang dideklarasikan di interface.

PERBEDAAN INTERFACE DAN ABSTRACT CLASS:

1. Interface hanya berisi method abstract (sebelum Java 8), sedangkan abstract class bisa punya method concrete dan abstract.

2. Class bisa implements multiple interface, tapi hanya bisa extends satu abstract class.

3. Interface tidak bisa punya constructor, abstract class bisa.

4. Interface digunakan untuk mendefinisikan capability (apa yang bisa dilakukan), abstract class digunakan untuk mendefinisikan common behavior (IS-A relationship).

5. Semua member di interface secara default public, sedangkan abstract class bisa punya member dengan berbagai access modifier.

KESIMPULAN:

Program ini mendemonstrasikan penggunaan interface sebagai kontrak dalam sistem pembayaran. Interface memberikan fleksibilitas dan maintainability karena memisahkan definisi (what) dari implementasi (how). Dengan interface, kita bisa dengan mudah menambahkan payment method baru (CreditCard, BankTransfer, dll) yang mengimplementasikan interface yang sama tanpa mengubah code yang sudah ada. Ini sesuai dengan prinsip Open/Closed Principle dalam SOLID design patterns: open for extension, closed for modification.