LAPORAN PRAKTIKUM MINGGU 5

Abstraction (Abstract Class & Interface)

NAMA : ALVIRA LIBRA RAMADHANI

NIM : 240202851

KELAS : 3IKRA

1. TUJUAN

1. Mahasiswa mampu menjelaskan perbedaan abstract class dan interface.
2. Mahasiswa mampu mendesain abstract class dengan method abstrak sesuai kebutuhan kasus.
3. Mahasiswa mampu membuat interface dan mengimplementasikannya pada class.
4. Mahasiswa mampu menerapkan multiple inheritance melalui interface pada rancangan kelas.
5. Mahasiswa mampu mendokumentasikan kode dengan komentar dan laporan.

2. DASAR TEORI

Abstraksi adalah proses menyederhanakan kompleksitas dengan menampilkan elemen penting dan menyembunyikan detail implementasi.

a. Abstract class: tidak dapat diinstansiasi, dapat memiliki method abstrak (tanpa badan) dan non-abstrak. Dapat menyimpan state (field).
b. Interface: kumpulan kontrak (method tanpa implementasi konkret). Sejak Java 8 mendukung default method. Mendukung multiple inheritance (class dapat mengimplementasikan banyak interface).
c. Gunakan abstract class bila ada *shared state* dan perilaku dasar; gunakan interface untuk mendefinisikan kemampuan/kontrak lintas hierarki.

Dalam konteks Agri-POS, Pembayaran dapat dimodelkan sebagai abstract class dengan method abstrak prosesPembayaran() dan biaya().
Implementasi konkritnya: Cash dan EWallet. Kemudian, interface seperti Validatable (mis. verifikasi OTP) dan Receiptable (mencetak bukti) dapat diimplementasikan oleh jenis pembayaran yang relevan.

3. LANGKAH PRAKTIKUM

1. Abstract Class – Pembayaran
   - Buat Pembayaran (abstract) dengan field invoiceNo, total dan method:
     a) double biaya() (abstrak) → biaya tambahan (fee).
     b) boolean prosesPembayaran() (abstrak) → mengembalikan status berhasil/gagal.
     c) double totalBayar() (konkrit) → return total + biaya();

2. Subclass Konkret
   - Cash → biaya = 0, proses = selalu berhasil jika tunai >= totalBayar().
   - EWallet → biaya = 1.5% dari total; proses = membutuhkan validasi.

3. Interface
   - Validatable → boolean validasi(); (contoh: OTP).
   - Receiptable → String cetakStruk();

4. Multiple Inheritance via Interface
   - EWallet mengimplementasikan **dua interface**: Validatable, Receiptable.
   - Cash setidaknya mengimplementasikan Receiptable.

5. Main Class
   - Buat MainAbstraction.java untuk mendemonstrasikan pemakaian Pembayaran (polimorfik).
   - Tampilkan hasil proses dan struk. Di akhir, panggil CreditBy.print("[NIM]", "[Nama]").

6. Commit dengan pesan: week5-abstraction-interface.

4. KODE PROGRAM

a) Receiptable.java

praktikum > week5-abstraction-interface > src > main > java > com > upb >

```java
package com.upb.agripos.model.kontrak;

public interface Receiptable {
    String cetakStruk();
}

```

b) Validatable.java

praktikum > week5-abstraction-interface > src > main > java > com > upb > agripos >

```java
package com.upb.agripos.model.kontrak;

public interface Validatable {
    boolean validasi(); // misal validasi OTP/PIN
}

```

c) Cash.java

praktikum > week5-abstraction-interface > src > main > java > com > upb > agripos > model > pembayaran >

```java
package com.upb.agripos.model.pembayaran;

import com.upb.agripos.model.kontrak.Receiptable;

public class Cash extends Pembayaran implements Receiptable {
    private double tunai;

    public Cash(String invoiceNo, double total, double tunai) {
        super(invoiceNo, total);
        this.tunai = tunai;
    }

    @Override
    public double biaya() {
        return 0.0;
    }

    @Override
    public boolean prosesPembayaran() {
        return tunai >= totalBayar();
    }

    @Override
    public String cetakStruk() {
        return String.format(
            format: "INVOICE %s | TOTAL: %.0f | BAYAR CASH: %.0f | KEMBALI: %.0f",
            invoiceNo, total, tunai, Math.max(a: 0, tunai - totalBayar())
        );
    }
```

d) EWallet.java

praktikum > week5-abstraction-interface > src > main > java > com > upb > agripos > model > pembayaran >

```java
package com.upb.agripos.model.pembayaran;

import com.upb.agripos.model.kontrak.Validatable;
import com.upb.agripos.model.kontrak.Receiptable;

public class EWallet extends Pembayaran implements Validatable, Receiptable {
    private String akun;
    private String otp;
    private static final double FEE = 2000;

    public EWallet(String invoiceNo, double total, String akun, String otp) {
        super(invoiceNo, total);
        this.akun = akun;
        this.otp = otp;
    }

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

    @Override
    public boolean validasi() {
        return otp != null && otp.length() == 6;
    }
```

```java
    @Override
    public boolean prosesPembayaran() {
        return validasi();
    }

    @Override
    public String cetakStruk() {
        return String.format(
            format: "INVOICE %s | TOTAL+FEE: %.0f (%.0f + %.0f) | E-WALLET: %s | STATUS: %s",
            invoiceNo, totalBayar(), total, biaya(), akun, prosesPembayaran() ? "BERHASIL" : "GAGAL"
        );
    }
}

```

e) Pembayaran.java

praktikum > week5-abstraction-interface > src > main > java > com > upb > agripos > model > pembayaran > J Per

```java
package com.upb.agripos.model.pembayaran;

public abstract class Pembayaran {
    protected String invoiceNo;
    protected double total;

    public Pembayaran(String invoiceNo, double total) {
        this.invoiceNo = invoiceNo;
        this.total = total;
    }

    public abstract double biaya();              // biaya tambahan/admin
    public abstract boolean prosesPembayaran();  // proses spesifik tiap metode

    public double totalBayar() {
        return total + biaya();
    }

    public String getInvoiceNo() { return invoiceNo; }
    public double getTotal() { return total; }
}
```

f) TranferBank.java

praktikum > week5-abstraction-interface > src > main > java > com > upb > agripos > model > pembayaran > J TransferBa

```java
package com.upb.agripos.model.pembayaran;

import com.upb.agripos.model.kontrak.Receiptable;
import com.upb.agripos.model.kontrak.Validatable;

public class TransferBank extends Pembayaran implements Validatable, Receiptable {
    private String akunBank;
    private String otp;
    private static final double BIAYA_TRANSFER = 2500;

    public TransferBank(String invoiceNo, double total, String akunBank, String otp) {
        super(invoiceNo, total);
        this.akunBank = akunBank;
        this.otp = otp;
    }

    @Override
    public double biaya() {
        return BIAYA_TRANSFER;
    }
```

g) CreditBy.java

```java
package com.upb.agripos.util;

public class CreditBy {
    private static final String NAMA = "Alviraa";
    private static final String NIM = "240202851";

    public static void print() {
        System.out.println("Credit by " + NAMA + " | NIM: " + NIM);
    }
}
```

h) MainAbtraction.java

```java
package com.upb.agripos;

import com.upb.agripos.model.kontrak.*;
import com.upb.agripos.model.pembayaran.*;
import com.upb.agripos.util.CreditBy;

public class MainAbstraction {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Pembayaran cash = new Cash(invoiceNo: "ALR-001", total: 200000, tunai: 250000);
        System.out.println(((Receiptable) cash).cetakStruk());

        Pembayaran ew = new EWallet(invoiceNo: "LBR-002", total: 221300, akun: "viraa@ewallet", otp: "020505");
        System.out.println(((Receiptable) ew).cetakStruk());

        Pembayaran transfer = new TransferBank(invoiceNo: "VRA-003", total: 200000, akunBank: "viraa@bank", otp: "112255");
        System.out.println(((Receiptable) transfer).cetakStruk());

        CreditBy.print();
    }
}
```

5. HASIL EKSEKUSI

```
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week5-abstraction-interface\src\main\java> javac com/upb/agripos/model
/pembayaran/*.java com/upb/agripos/model/kontrak/*.java com/upb/agripos/util/*.java com/upb/agripos/*.java
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week5-abstraction-interface\src\main\java> java com.upb.agripos.MainAb
straction
INVOICE ALR-001 | TOTAL: 200000 | BAYAR CASH: 250000 | KEMBALI: 50000
INVOICE LBR-002 | TOTAL+FEE: 223300 (221300 + 2000) | E-WALLET: viraa@ewallet | STATUS: BERHASIL
INVOICE VRA-003 | TOTAL+BIAYA: 202500 (200000 + 2500) | TRANSFER BANK: viraa@bank | STATUS: BERHASIL
Credit by Alviraa | NIM: 240202851
PS C:\Users\LENOVO\oop-202501-240202851\praktikum\week5-abstraction-interface\src\main\java>
```

- Analisis
  Pada praktikum ini diterapkan konsep **abstraction** dan **interface** untuk memisahkan logika program dan menentukan perilaku yang harus dimiliki oleh setiap kelas. Penggunaan *abstract class* dan *interface* membuat struktur program lebih terorganisir, fleksibel, serta mudah dikembangkan.

- Kendala dan Solusi
  a) **Kendala:** Error saat compile karena penulisan path .java tidak benar.
  Solusi: Gunakan *.java untuk mengompilasi semua file dalam folder.
  b) **Kendala:** ClassNotFoundException saat menjalankan program.
  Solusi: Pastikan menggunakan -d bin saat compile dan java -cp bin com.upb.agripos.MainAbstraction saat run.

c) **Kendala:** Bingung membedakan *abstract class* dan *interface.*
Solusi: Pahami bahwa *abstract class* boleh memiliki implementasi, sedangkan *interface* hanya mendefinisikan kontrak.

6. KESIMPULAN

Melalui praktikum ini, dapat disimpulkan bahwa penggunaan **abstraction** dan **interface** membantu membuat program lebih terstruktur, efisien, dan mudah dikembangkan. Dengan memisahkan definisi perilaku dan implementasinya, kode menjadi lebih fleksibel serta mendukung prinsip dasar pemrograman berorientasi objek seperti *modularity* dan *reusability*.

7. CHEKLIST KEBERHASILAN

☑Abstract class Pembayaran memiliki method abstrak dan method konkrit yang tepat.
☑Interface diimplementasikan dengan benar pada kelas yang relevan.
☑Multiple inheritance via interface berjalan (kelas mengimplementasikan ≥2 interface).
☑Program menampilkan struk dan status proses pembayaran.
☑Output menyertakan credit by: [NIM] - [Nama] melalui CreditBy.
☑Screenshot & laporan telah dilampirkan.

8. QUIZ

1. Jelaskan perbedaan konsep dan penggunaan **abstract class** dan **interface**.
   Jawaban:
   - **Abstract class** digunakan untuk mendefinisikan kelas dasar yang memiliki sebagian implementasi dan sebagian lainnya masih abstrak. Kelas ini dapat memiliki atribut, konstruktor, dan metode dengan isi.
   - **Interface** digunakan untuk mendefinisikan kontrak atau perilaku yang harus dimiliki oleh kelas lain tanpa memberikan implementasi sama sekali. Interface hanya berisi deklarasi metode (dan konstanta).
     Perbedaan utamanya: abstract class mewarisi dengan extends (hanya satu), sedangkan interface diimplementasikan dengan implements (bisa lebih dari satu).

2. Mengapa **multiple inheritance** lebih aman dilakukan dengan interface pada Java?
   Jawaban:
   Karena interface **tidak membawa implementasi atau state (variabel instance)**, sehingga tidak menimbulkan konflik pewarisan seperti pada multiple inheritance di class (misalnya dua parent memiliki method dengan nama sama). Dengan interface, Java hanya mewarisi kontrak perilaku, bukan isi kode, sehingga aman dan tidak menimbulkan ambiguitas.

3. Pada contoh Agri-POS, bagian mana yang **paling tepat** menjadi abstract class dan mana yang menjadi interface? Jelaskan alasannya.
   Jawaban:

- **Abstract class:** Pembayaran → karena memiliki atribut dan logika umum (seperti jumlah, tanggal, metode bayar) yang bisa digunakan ulang oleh kelas turunan seperti CashPayment atau EWalletPayment.
- **Interface:** Validatable → karena hanya mendefinisikan perilaku (misalnya validate()) yang wajib dimiliki oleh beberapa kelas tanpa perlu mengetahui bagaimana validasi dilakukan.
  Dengan cara ini, Pembayaran berfungsi sebagai dasar struktur umum, sementara Validatable menjadi kontrak perilaku yang dapat diterapkan di berbagai kelas yang membutuhkan validasi.