**LAPORAN PRAKTIKUM**

**Abstract Class & Interface**



**Disusun Oleh:**

**Dzakir Tsabit Asy Syafiq (241511071)**

**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika**

**Politeknik Negeri Bandung**

**17/09/2025**

Source Code :

import java.util.Scanner;

public class PujasPolban {

    private String nama;

    private int jumlah;

    private int harga;

    public PujasPolban(String nama, int jumlah, int harga)

    {

        this.nama = nama;

        this.jumlah = jumlah;

        this.harga = harga;

    }

    public String GetNama()

    {

        return nama;

    }

    public int Getjumlah()

    {

        return jumlah;

    }

    public int GetHarga()

    {

        return harga;

    }

    // public void SetNama(String newnama)

    // {

    //     this.nama = newnama;

    // }

    // public void Setjumlah(int newjumlah)

    // {

    //     this.jumlah = newjumlah;

    // }

    // public void SetHarga(int newharga)

    // {

    //     this.harga = newharga;

    // }

    public static void main(String []args) throws java.io.IOException {

        Scanner input = new Scanner(System.in);

        System.out.println("-- MENU MAKANAN PUJAS --");

        System.out.println("1. Nasi Goreng Spesial");

        System.out.println("2. Mie Goreng Spesial");

        System.out.println("3. Ayam Bakar Spesial");

        System.out.println("4. Ayam Geprek Spesial");

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

        System.out.print("Pilihan Anda : ");

        int pilihan = input.nextInt();

        String nama\_makan = "";

        int harga\_mkn =0;

        switch (pilihan) {

            case 1: nama\_makan = "Nasi Goreng Spesial"; harga\_mkn = 15000; break;

            case 2: nama\_makan = "Mie Goreng Spesial"; harga\_mkn = 10000; break;

            case 3: nama\_makan = "Ayam Bakar Spesial"; harga\_mkn = 20000; break;

            case 4: nama\_makan = "Ayam Geprek Spesial"; harga\_mkn = 12000; break;

        }

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

        int jml\_mkn = input.nextInt();

        PujasPolban pesanan = new PujasPolban(nama\_makan, jml\_mkn, harga\_mkn);

        int total = pesanan.GetHarga() \* jml\_mkn;

        System.out.println("\n-- STRUK PESANAN --");

        System.out.println("Pesanan   : " + pesanan.GetNama());

        System.out.println("Jumlah    : " + pesanan.Getjumlah());

        System.out.println("Harga     : Rp" + pesanan.GetHarga());

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

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

    }

}

**1. Analisis kode**

* Program sederhana untuk pemesanan makanan lewat console.
* Kelas PujasPolban menyimpan atribut nama, jumlah, harga; menyediakan konstruktor dan getter.
* main() menampilkan menu, membaca pilihan dan jumlah, membuat objek PujasPolban, menghitung total, lalu mencetak struk.
* Kelemahan/area perbaikan:
  + Tidak ada penanganan input invalid (pilihan diluar 1–4 atau input non-angka).
  + Getter menggunakan konvensi nama campuran (GetNama, Getjumlah — sebaiknya getNama(), getJumlah()).
  + Atribut jumlah dan harga bersifat int; untuk keuangan lebih aman pakai long atau BigDecimal bila butuh presisi.
  + Tidak ada pemisahan tanggung jawab (IO di main, model di kelas). Bisa ditambahkan kelas layanan (Service) untuk logika pemesanan.
  + Tidak ada komentar maupun test.\

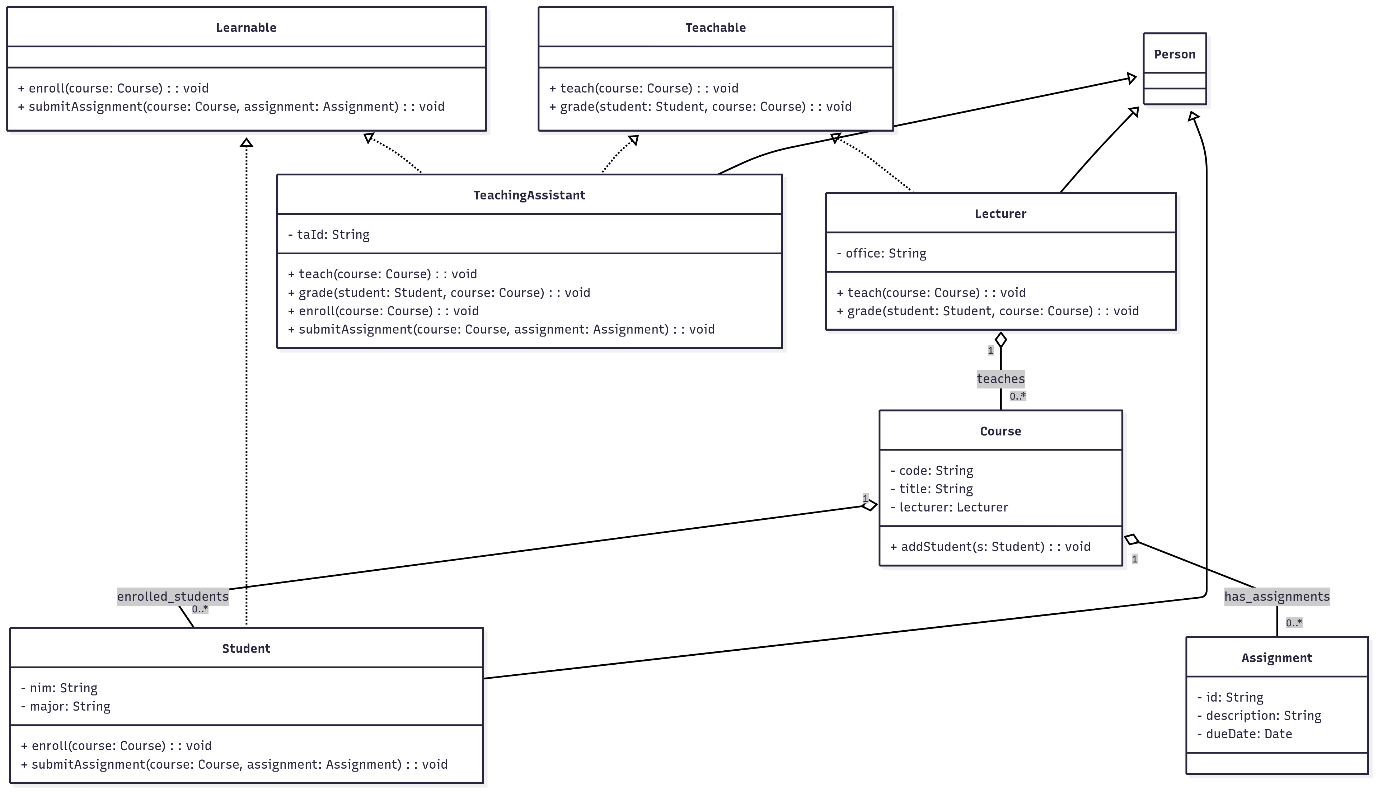
1. **Di bagian mana diterapkan interface dan multiple inheritance**

* Kode saya tidak memiliki interface ataupun inheritance
* Jika ingin menambahkan interface dan multiple inheritance (di Java: kelas dapat meng-extend satu abstract/class dan mengimplementasikan satu atau lebih interface), Anda bisa:
  + Tambahkan sebuah abstract class (mis. MenuItemBase) yang menyediakan implementasi dasar/field umum (nama, harga) dan/atau metode abstrak.
  + Tambahkan interface (mis. Orderable atau Printable) yang mendefinisikan perilaku seperti getPrice(), getName(), printReceipt().
  + Buat kelas PujasPolban (atau kelas baru, mis. MenuItem atau Pesanan) yang extends abstract class dan implements satu atau lebih interface — ini memberi contoh multiple inheritance via class + interface\

1. **Solusi minimal menggunakan 1 abstract class dan 1 interface Contoh struktur minimal:**

* Abstract class MenuItemBase
  + fields: String nama; int harga;
  + constructor, concrete getter untuk nama/harga, mungkin method abstrak getType() atau calculatePrice(int qty).
* Interface Orderable
  + method signatures: int getHarga(); String getNama(); int getJumlah(); int calculateTotal();
* Kelas Pesanan extends MenuItemBase implements Orderable
  + menambahkan field jumlah; mengimplementasikan calculateTotal() dan getter lainnya. Implementasi singkat (pseudo-Java):
* abstract class MenuItemBase { protected String nama; protected int harga; public MenuItemBase(String n,int h){...} public String getNama(){...} public int getHarga(){...} public abstract String getType(); }
* interface Orderable { int getJumlah(); int calculateTotal(); }
* class Pesanan extends MenuItemBase implements Orderable { private int jumlah; ... public int getJumlah(){...} public int calculateTotal(){ return getHarga()\*jumlah; } public String getType(){ return "Makanan"; } }

1. Diagram kelas kasus akademik yang menerapkan interface dan multiple inheritance Saya akan memberikan deskripsi diagram kelas :



* Abstract class Person
  + -id: String
  + -name: String
  + +Person(id:String, name:String)
  + +getId():String
  + +getName():String
* Interface Teachable
  + +teach(course:Course):void
  + +grade(student:Student, course:Course):void
* Interface Learnable
  + +enroll(course:Course):void
  + +submitAssignment(course:Course, assignment:Assignment):void
* Class Lecturer extends Person implements Teachable
  + -office:String
  + +teach(course:Course):void
  + +grade(student:Student, course:Course):void
* Class Student extends Person implements Learnable
  + -nim:String
  + -major:String
  + +enroll(course:Course):void
  + +submitAssignment(course:Course, assignment:Assignment):void
* Class TeachingAssistant extends Person implements Teachable, Learnable
  + -taId:String
  + +teach(course:Course):void
  + +grade(student:Student, course:Course):void
  + +enroll(course:Course):void
  + +submitAssignment(course:Course, assignment:Assignment):void
  + (TeachingAssistant menunjukkan multiple interface implementation — gabungan peran dosen dan mahasiswa)
* Class Course
  + -code:String
  + -title:String
  + -lecturer:Lecturer
  + +addStudent(s:Student):void
* Class Assignment
  + -id:String
  + -description:String
  + -dueDate:Date

Relasi:

* Person <- Lecturer (extends)
* Person <- Student (extends)
* Person <- TeachingAssistant (extends)
* Lecturer implements Teachable
* Student implements Learnable
* TeachingAssistant implements Teachable and Learnable (multiple inheritance via interfaces)
* Course has Lecturer (aggregation) and list of Students
* Assignment associated with Course