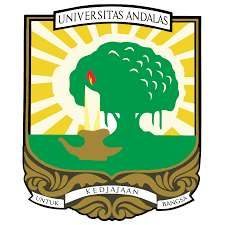
**LAPORAN PRAKTIKUM PBO**

**ABSTRACT CLASS, INTERFACE, FINAL CLASS, INNER CLASS**

****

**Disusun Oleh:**

**INDAH SYAHFITRI**

**2311532016**

**Dosen Pembimbing: Afdhal Dinilhak, S.Komp., M.Kom.**

**DEPARTEMEN INFORMATIKA**

**FAKULTAS TEKNOLOGI INFORMASI**

**UNIVERSITAS ANDALAS**

1. **PENDAHULUAN**

Pemrograman berorientasi objek (OOP) adalah pemrograman yang banyak digunakan untuk merancang dan mengembangkan perangkat lunak dengan menggunakan konsep objek dan kelas. Dalam OOP, terdapat beberapa konsep lanjutan yang penting untuk dipahami agar dapat mengembangkan program yang lebih modular dan terstruktur. Konsep-konsep tersebut meliputi Abstract Class, Interface, Final Class, dan Inner Class.

Abstract Class adalah kelas yang tidak dapat diinstansiasi langsung dan digunakan sebagai kelas dasar untuk kelas lain yang lebih spesifik. Abstract class memungkinkan kita untuk mendeklarasikan metode yang tidak memiliki implementasi, yang kemudian diimplementasikan oleh kelas turunan. Interface adalah tipe data abstrak yang hanya dapat berisi deklarasi metode dan konstan. Sebuah kelas dapat mengimplementasikan satu atau lebih interface untuk mendefinisikan perilaku yang diinginkan tanpa bergantung pada hierarki pewarisan.

Final Class adalah kelas yang tidak dapat diturunkan oleh kelas lain. Penggunaan kelas final dapat membantu untuk membuat kelas yang tidak boleh diubah atau dimodifikasi oleh subclass. Inner Class adalah kelas yang didefinisikan di dalam kelas lain. Inner class dapat digunakan untuk menyederhanakan struktur kode dan membatasi akses ke kelas-kelas yang hanya digunakan dalam konteks tertentu.Dalam praktikum ini, mahasiswa akan mempelajari implementasi dan penggunaan masing-masing konsep tersebut dalam sebuah contoh kasus, yakni Perusahaan Transportasi X ingin membuat aplikasi untuk mengelola berbagai jenis kendaraan mereka, seperti mobil, sepeda motor, dan bus. Setiap kendaraan memiliki karakteristik yang berbeda, namun beberapa atribut/properti dan method bisa dikategorikan dan diatur dalam struktur yang sama.

1. **TUJUAN PRAKTIKUM**
2. Mahasiswa mampu memahami Konsep Abstract Class.
3. Mahasiswa mampu mengimplementasikan Interface dalam Java.
4. Mahasiswa mampu menggunakan Final Class.
5. Mahasiswa mampu menggunakan Inner Class dalam Program.
6. **LANGKAH-LANGKAH**
7. Buat package dengan nama com.informatika.application dan com.informatika data. Lalu, pada package com.informatika.data, buatlah sebuah abstract class dengan nama Kendaraan.

Sebuah gambar berisi teks, cuplikan layar, tampilan, software

Deskripsi dibuat secara otomatis

Class ini memiliki properti merk, model, dan tahunProduksi kendaraan dan memiliki metode abstrak nyalakanMesin yang menunjukkan bagaimana mesin kendaraan dinyalakan. Juga, tambahkan final method tampilkanInfo untuk menampilkan informasi dasar kendaraan (merk, model, dan tahun produksi). Di sini, kelas ini menggunakan constructor sebagai inisialisasi awal property yang akan digunakan. Abstract voidd adalah method yang bisa digunakan di kelas lain untuk berbagai kendaraan, namun output yang dikeluarkan akan disesuaikan nanti tergantung kendaraan.

1. Selanjutnya, buat kelas interface BahanBakar yang memiliki metode jenisBahanBakar yang mengembalikan jenis bahan bakar kendaraan (misalnya bensin, listrik) berdasarkan kendaraan tertentu. Dan tambahkan default method infoKonsumsi untuk menampilkan informasi umum mengenai konsumsi bahan bakar (misalnya, "Konsumsi bahan bakar tergantung kapasitas mesin").

Sebuah gambar berisi teks, software, Software multimedia, cuplikan layar

Deskripsi dibuat secara otomatis

1. Buat interface TransportasiUmum yang memperluas/mengikuti (extends) kelas interface BahanBakar sebelumnya dan tambahkan metode kapasitasPenumpang yang mengembalikan jumlah penumpang yang dapat diangkut oleh kendaraan tertentu.

Sebuah gambar berisi teks, cuplikan layar, Font, software

Deskripsi dibuat secara otomatis

1. Buat final class Mobil yang merupakan turunan dari Kendaraan dan mengimplementasikan kelas interface BahanBakar. Awalnya, mobil diinisialisasi dengan menggunakan atribut kelas Kendaraan. Properti tambahan untuk mobil adalah jenisTransmisi dalam bentuk String. Implementasikan metode nyalakanMesin untuk menunjukkan cara menyalakan mesin mobil. Implementasikan metode jenisBahanBakar dengan membuat perintah output jenis bahan bakar sesuai kendaraan tertentu. Tambahkan metode fiturMobil untuk menunjukkan fitur tambahan yang dimiliki oleh mobil tersebut seperti contoh berikut.

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Deskripsi dibuat secara otomatis

1. Buat class Bus yang merupakan turunan dari Kendaraan dan mengimplementasikan TransportasiUmum. Awalnya, bus diinisialisasi dengan menggunakan atribut kelas Kendaraan. Properti tambahan untuk bus adalah kelasBus. Implementasikan metode nyalakanMesin untuk menunjukkan cara menyalakan mesin bus. Implementasikan metode jenisBahanBakar dan kapasitasPenumpang sebagai informasi jumlah penumpang yang dapat dimuat bus. Tambahkan metode fiturBus untuk menunjukkan fitur tambahan yang dimiliki oleh bus tersebut seperti contoh.

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Deskripsi dibuat secara otomatis

Tambahkan inner class JadwalPerjalanan di dalam class Bus dengan properti rute dan waktuBerangkat. Buat menggunakan constructor untuk inisialisasi, lalu buat method untuk menampilkan informasi jadwal perjalanan bus, yakni tampilkanJadwal().

1. Setelah selesai, buat kelas com.informatika.application lalu panggil kelas Mobil dan bus. Buat inisialisasi nama mobil ataupun bus lalu buat seperti berikut untuk mengisi atribut yang telah dibuat sebelumnya, seperti merk, model, dan tahun produksi. Serta, panggil method yang telah dibuat sebelumnya seperti jenis bahan bakar, tampilkan info, nyalakan mesin, dll.

Sebuah gambar berisi teks, cuplikan layar, software, Software multimedia

Deskripsi dibuat secara otomatis

1. Jika dieksekusi, maka akan muncul output seperti ini.

Sebuah gambar berisi teks, cuplikan layar, Font

Deskripsi dibuat secara otomatis

1. Selanjutnya, buat interface TransportasiUdara yang memperluas/menurunkan sifat (extends) interface BahanBakar. Tambahkan metode jenisPenerbangan yang menampilkan informasi jenis penerbangan dari transportasi udara tersebut dengan menggunakan void seperti berikut.

Sebuah gambar berisi teks, cuplikan layar, Font, garis

Deskripsi dibuat secara otomatis

1. Buat interface Maskapai yang memiliki metode namaMaskapai yang mengembalikan nama maskapai dari pesawat.

**Sebuah gambar berisi teks, cuplikan layar, Font, nomor

Deskripsi dibuat secara otomatis**

1. Buat class Pesawat yang merupakan turunan dari Kendaraan dan mengimplementasikan interface TransportasiUdara dan Maskapai. Inisialisasi dengan menggunakan constructor. Lalu, panggil method nyalakan mesin dengan output sesuai pesawat. Panggil jenis bahan bakar dengan return. Panggil nama maskapai dengan return dan jenis penerbangan dengan membuat output yang sesuai dengan pesawat.

Sebuah gambar berisi teks, cuplikan layar, software, tampilan

Deskripsi dibuat secara otomatis

1. Setelah itu, lengkapi kelas KendaraanApp pada package com.informatika.application dengan memanggil kelas pesawat. Buat inisialisasi nama pesawat lalu buat seperti berikut untuk mengisi atribut yang telah dibuat sebelumnya, seperti merk, model, dan tahun produksi. Serta, panggil method yang telah dibuat sebelumnya seperti jenis bahan bakar, tampilkan info, dll.

Sebuah gambar berisi teks, cuplikan layar, Font, garis

Deskripsi dibuat secara otomatis

1. Jika sudah, maka outputnya akan seperti ini jika program dieksekusi.

Sebuah gambar berisi teks, Font, cuplikan layar

Deskripsi dibuat secara otomatis

1. **SIMPULAN**

Praktikum ini bertujuan untuk merancang aplikasi pengelolaan kendaraan menggunakan konsep *Object-Oriented Programming* (OOP). Aplikasi ini mengelola berbagai jenis kendaraan, seperti mobil, bus, dan pesawat, dengan memanfaatkan prinsip-prinsip OOP seperti pewarisan, interface, dan kelas abstrak. Kelas Kendaraan berfungsi sebagai kelas dasar dengan properti umum seperti merk, model, dan tahun produksi, serta metode abstrak nyalakanMesin() yang diimplementasikan oleh setiap subclass kendaraan.

Dalam praktikum ini, dibuat beberapa interface untuk memberikan fleksibilitas pada kendaraan. Interface BahanBakar mengatur metode jenisBahanBakar() untuk mengetahui jenis bahan bakar yang digunakan kendaraan, sementara TransportasiUmum yang memperluas BahanBakar menambahkan metode kapasitasPenumpang() untuk kendaraan yang digunakan dalam transportasi umum, seperti bus. Setiap kendaraan, seperti Mobil dan Bus, mengimplementasikan interface yang sesuai dan memiliki metode tambahan untuk fitur khusus kendaraan tersebut.

Aplikasi ini juga mendemonstrasikan penggunaan inner class dengan membuat JadwalPerjalanan dalam kelas Bus untuk mengelola informasi jadwal perjalanan. Kelas Pesawat mengimplementasikan interface TransportasiUdara dan Maskapai untuk mengelola informasi penerbangan dan maskapai. Secara keseluruhan, praktikum ini berhasil mengilustrasikan penerapan OOP untuk membuat aplikasi yang dapat mengelola berbagai jenis kendaraan dengan struktur yang terorganisir dan mudah diperluas.