STUDI KASUS CB2 PEMROGRAMAN DASAR

Penerapan Abstract Class dalam Java

KELOMPOK 6

NAMA	NIM
1. Muhammad Hafiizh Anindya	255150207111057
2. Ulfa Aulia Sakina	255150207111058
3. Aryan Zaky Prayogo	255150207111059
4. Faeyza Safa Izz Deyardi	255150207111060
5. Raven Ravellyn Sulistyo	255150207111061
6. Jaler Zuhdi Nail	255150207111062

I. Latar Belakang

Studi kasus ini dirancang untuk memahami konsep abstract class dalam bahasa pemrograman Java sebagai bagian dari pembelajaran Object-Oriented Programming (OOP). Abstract class digunakan sebagai kerangka dasar (template) yang tidak dapat diinstansiasi langsung, tetapi berfungsi untuk mewariskan struktur dan perilaku dasar kepada subclass-nya.

Dalam studi kasus ini, mahasiswa diminta membuat simulasi sistem rental kendaraan sederhana dengan berbagai jenis kendaraan (mobil, motor, dan truk). Setiap kendaraan memiliki kesamaan atribut seperti biaya sewa harian dan status ketersediaan, tetapi memiliki cara berbeda dalam

menghitung biaya tambahan. Tugas ini bertujuan agar mahasiswa memahami cara membuat dan menggunakan abstract class, hubungan antar superclass dan subclass, serta penerapan inheritance dan polymorphism.

II. Tujuan Pembelajaran

1. Memahami peran abstract class sebagai superclass yang tidak bisa diinstansiasi langsung.

2. Menerapkan method abstract untuk memaksa subclass mengimplementasikan logika spesifik.

3. Mengintegrasikan konsep inheritance agar menghindari duplikasi kode.

4. Melatih debugging, testing, dan kolaborasi tim dalam pengembangan program.

III. Deskripsi Masalah

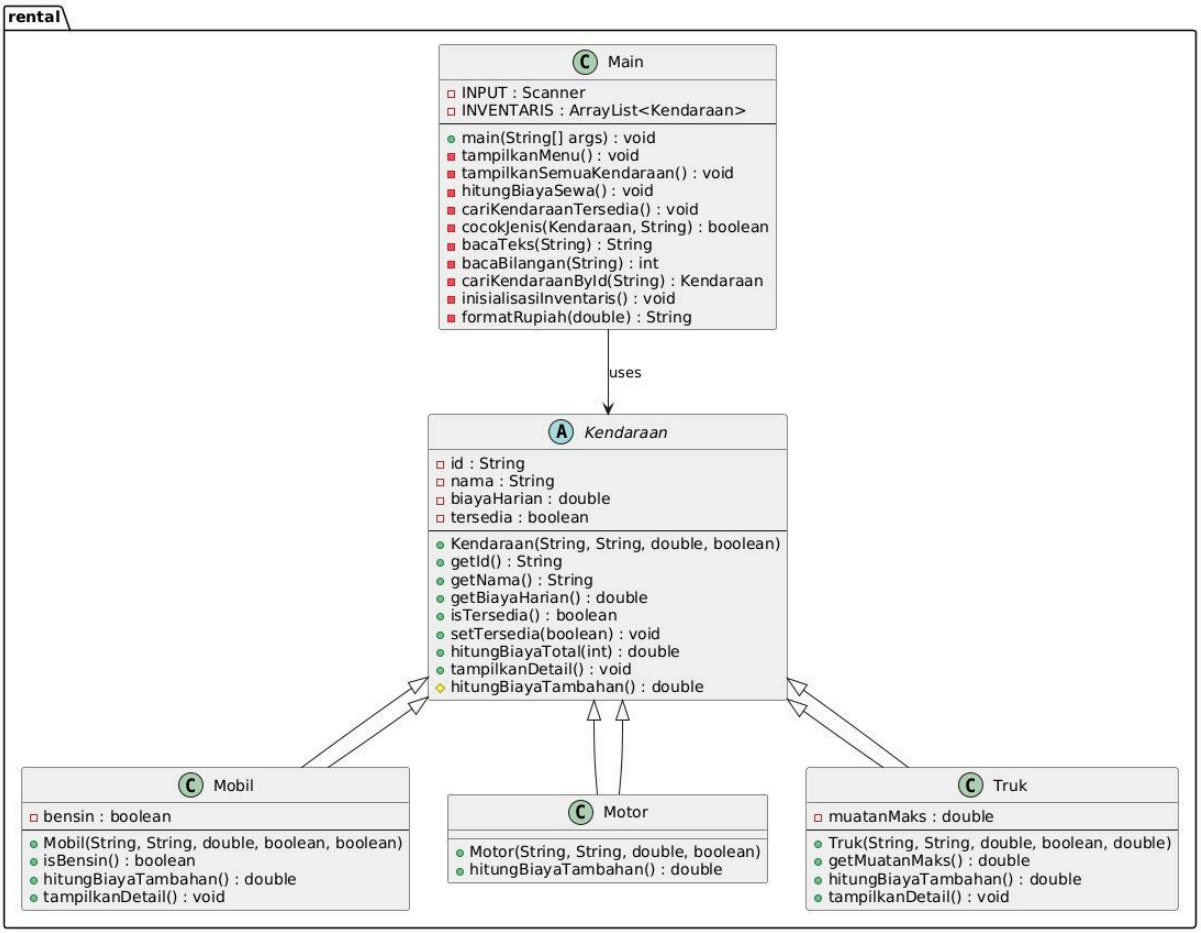
Perusahaan rental kendaraan memerlukan sistem untuk mengelola inventaris kendaraan. Setiap kendaraan memiliki ID, nama/model, biaya sewa harian, dan status ketersediaan. Perhitungan biaya tambahan berbeda untuk setiap jenis kendaraan:

Jenis Kendaraan	Ketentuan Biaya Tambahan
Mobil	Tambahan 10% dari biaya dasar jika berbahan bakar bensin.
Motor	Tambahan tetap Rp 20.000 per hari.
Truk	Tambahan Rp 50.000 per ton muatan maksimal.

IV. Perancangan Program

Struktur class terdiri atas satu abstract class Kendaraan dan tiga subclass: Mobil, Motor, dan Truk.

Diagram hubungan antar class: Kendaraan (abstract) → Mobil, Motor, Truk



V. Implementasi dan Kode Program

Berikut kode utama:

```
package rental;


import java.util.ArrayList;
import java.util.Scanner;


public class Main {
    private static final Scanner INPUT = new Scanner(System.in);
    private static final ArrayList<Kendaraan> INVENTARIS = new ArrayList<>();


    public static void main(String[] args) {
        inisialisasiInventaris();
        boolean berjalan = true;
        while (berjalan) {
            tampilkanMenu();
            String pilihan = bacaTeks("Pilih menu: ");
            switch (pilihan) {
                case "1":
                    tampilkanSemuaKendaraan();
                    break;
                case "2":
                    hitungBiayaSewa();
                    break;
                case "3":
                    cariKendaraanTersedia();
```

```
                break;
            case "0":
                berjalan = false;
                System.out.println("Terima kasih, sampai jumpa!");
                break;
            default:
                System.out.println("Pilihan tidak dikenal. Silakan coba lagi.");
        }
    }
    INPUT.close();
}
private static void tampilkanMenu() {
    System.out.println();
    System.out.println("==== Sistem Rental Kendaraan ====");
    System.out.println("1. Tampilkan semua kendaraan");
    System.out.println("2. Hitung biaya sewa kendaraan");
    System.out.println("3. Cari kendaraan tersedia berdasarkan jenis");
    System.out.println("0. Keluar");
}
private static void tampilkanSemuaKendaraan() {
    System.out.println("\n-- Daftar Kendaraan --");
    for (Kendaraan kendaraan : INVENTARIS) {
        kendaraan.tampilkanDetail();
    }
}
```

```
    private static void hitungBiayaSewa() {
        String id = bacaTeks("Masukkan ID kendaraan: ");
        Kendaraan kendaraan = cariKendaraanById(id);
        if (kendaraan == null) {
            System.out.println("Kendaraan dengan ID " + id + " tidak ditemukan.");
            return;
        }
        try {
            int hari = bacaBilangan("Masukkan jumlah hari sewa (1-7): ");
            double total = kendaraan.hitungBiayaTotal(hari);
            System.out.println("Biaya total untuk " + kendaraan.getNama() + " ID: " +
kendaraan.getId()
                    + " selama " + hari + " hari: " + formatRupiah(total));

            if (kendaraan.isTersedia()) {
                String konfirmasi = bacaTeks("Update status menjadi tidak tersedia? (y/n): ");
                if (konfirmasi.equalsIgnoreCase("y")) {
                    kendaraan.setTersedia(false);
                    System.out.println("Status kendaraan diperbarui menjadi tidak tersedia.");
                }
            } else {
                System.out.println("Catatan: kendaraan saat ini tidak tersedia.");
            }
        } catch (IllegalArgumentException ex) {
            System.out.println("Terjadi kesalahan: " + ex.getMessage());
```

```
    }
}
private static void cariKendaraanTersedia() {
    String jenis = bacaTeks("Masukkan jenis kendaraan (Mobil/Motor/Truk): ");
    String jenisLower = jenis.trim().toLowerCase();
    boolean ditemukan = false;

    for (Kendaraan kendaraan : INVENTARIS) {
        if (kendaraan.isTersedia() && cocokJenis(kendaraan, jenisLower)) {
            kendaraan.tampilkanDetail();
            ditemukan = true;
        }
    }
    if (!ditemukan) {
        System.out.println("Tidak ada kendaraan tersedia untuk jenis: " + jenis);
    }
}
private static boolean cocokJenis(Kendaraan kendaraan, String jenisLower) {
    switch (jenisLower) {
        case "mobil":
            return kendaraan instanceof Mobil;
        case "motor":
            return kendaraan instanceof Motor;
        case "truk":
        case "truck":
```

```
            return kendaraan instanceof Truk;
        default:
            return false;
    }
}
private static String bacaTeks(String prompt) {
    System.out.print(prompt);
    return INPUT.nextLine().trim();
}
private static int bacaBilangan(String prompt) {
    while (true) {
        String masukan = bacaTeks(prompt);
        try {
            return Integer.parseInt(masukan);
        } catch (NumberFormatException ex) {
            System.out.println("Input harus berupa angka. Silakan coba lagi.");
        }
    }
}
private static Kendaraan cariKendaraanById(String id) {
    for (Kendaraan kendaraan : INVENTARIS) {
        if (kendaraan.getId().equalsIgnoreCase(id)) {
            return kendaraan;
        }
    }
```

```
            return null;
        }
        private static void inisialisasiInventaris() {
            INVENTARIS.add(new Mobil("MBS001", "Porsche 911 GT3", 650_000, true,
    true));
            INVENTARIS.add(new Mobil("MBL002", "McLaren 765s", 750_000, true, false));
            INVENTARIS.add(new Motor("MTS101", "BMW-S1000RR", 550_000, true));
            INVENTARIS.add(new Motor("MTS102", "CBR-1000RRR", 530_000, false));
            INVENTARIS.add(new Truk("TRK55", "Hino Dutro", 700_000, true, 4));
        }
        private static String formatRupiah(double angka) {
            long pembulatan = Math.round(angka);
            return "Rp " + pembulatan;
        }
    }
```

VI. Pengujian Program

Contoh output:

```
==== Sistem Rental Kendaraan ====
1. Tampilkan semua kendaraan
2. Hitung biaya sewa kendaraan
3. Cari kendaraan tersedia berdasarkan jenis
0. Keluar
Pilih menu:
1
-- Daftar Kendaraan --
ID: MBS001, Nama: Porsche 911 GT3, Biaya Harian: Rp 650.000, Status: Tersedia
  Tipe Bahan Bakar: Bensin
ID: MBL002, Nama: McLaren 765s, Biaya Harian: Rp 750.000, Status: Tersedia
  Tipe Bahan Bakar: Non-bensin
Jenis: Motor
```

```
ID: MTS101, Nama: BMW-S1000RR, Biaya Harian: Rp 550.000, Status: Tersedia
Helm disediakan: Ya
Jenis: Motor
ID: MTS102, Nama: CBR-1000RRR, Biaya Harian: Rp 530.000, Status: Tidak Tersedia
Helm disediakan: Ya
ID: TRK55, Nama: Hino Dutro, Biaya Harian: Rp 700.000, Status: Tersedia
  Muatan Maksimum: 4.0 ton

==== Sistem Rental Kendaraan ====
1. Tampilkan semua kendaraan
2. Hitung biaya sewa kendaraan
3. Cari kendaraan tersedia berdasarkan jenis
0. Keluar
Pilih menu:
2
Masukkan ID kendaraan: MBS001
Masukkan jumlah hari sewa (1-7):
5
Biaya total untuk Porsche 911 GT3 ID: MBS001 selama 5 hari: Rp 3315000
Update status menjadi tidak tersedia? (y/n):
y
Status kendaraan diperbarui menjadi tidak tersedia.

==== Sistem Rental Kendaraan ====
1. Tampilkan semua kendaraan
2. Hitung biaya sewa kendaraan
3. Cari kendaraan tersedia berdasarkan jenis
0. Keluar
Pilih menu:
2
Masukkan ID kendaraan:
MTS102
Masukkan jumlah hari sewa (1-7):
3
Biaya total untuk CBR-1000RRR ID: MTS102 selama 3 hari: Rp 1610000
Catatan: kendaraan saat ini tidak tersedia.

==== Sistem Rental Kendaraan ====
1. Tampilkan semua kendaraan
2. Hitung biaya sewa kendaraan
3. Cari kendaraan tersedia berdasarkan jenis
0. Keluar
Pilih menu:
3
Masukkan jenis kendaraan (Mobil/Motor/Truk):
Motor
Jenis: Motor
ID: MTS101, Nama: BMW-S1000RR, Biaya Harian: Rp 550.000, Status: Tersedia
Helm disediakan: Ya

==== Sistem Rental Kendaraan ====
```

```
1. Tampilkan semua kendaraan
2. Hitung biaya sewa kendaraan
3. Cari kendaraan tersedia berdasarkan jenis
0. Keluar
Pilih menu:
0
Terima kasih, sampai jumpa!
```

VII. Kesimpulan

Abstract class efektif digunakan untuk menyediakan struktur dasar yang dapat diwariskan ke subclass. Konsep polymorphism membuat kode lebih fleksibel, dan inheritance mengurangi duplikasi kode. Tugas ini membantu memahami penerapan konsep OOP dalam sistem sederhana seperti manajemen rental kendaraan.

Link repository Github: Link Repository Github