LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK
MODUL 9
Overriding and Polymorphizm

Disusun Oleh:

Gerald Eberhard

(105223002)

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN KOMPUTER

UNIVERSITAS PERTAMINA

2025

1. Pendahuluan

Praktikum ini bertujuan untuk mengembangkan sistem pemesanan tiket transportasi sederhana menggunakan konsep pemrograman berorientasi objek (PBO) dalam bahasa pemrograman Java. Sistem ini dirancang untuk mengelola pemesanan tiket untuk tiga jenis transportasi: bus, kereta, dan pesawat. Setiap jenis transportasi memiliki informasi umum seperti nama, jumlah kursi, dan tujuan, serta perhitungan harga tiket yang berbeda berdasarkan jenis transportasi dan kelas layanan (Ekonomi, Bisnis, atau VIP). Konsep PBO yang diterapkan meliputi pewarisan, overriding, overloading, dan polimorfisme, yang memungkinkan pembuatan sistem yang modular, fleksibel, dan mudah dikembangkan. Tujuan utama praktikum ini adalah untuk memahami dan mengimplemen tasikan konsep-konsep tersebut dalam konteks nyata.

2. Variabel

No	Nama Variabel	Tipe Data	Fungsi
1	t	Transportasi	Variabel perulangan dalam loop for-each yang digunakan untuk mengakses setiap elemen dalam transportasiArray.
2	kelasLayanan	String	Variabel sementara untuk menyimpan nama kelas layanan (Ekonomi, Bisnis, VIP) yang digunakan dalam perhitungan harga tiket.
3	transportasiArray	Transportasi[]	Array yang menyimpan objek dari kelas Transportasi (termasuk Bus, Kereta, Pesawat) untuk implementasi polimorfisme.
4	pesawat	Transportasi[]	Objek dari kelas Pesawat yang mewakili transportasi pesawat dengan properti tertentu (nama, jumlahKursi, tujuan).
5	kereta	Transportasi[]	Objek dari kelas Kereta yang mewakili transportasi kereta dengan properti tertentu (nama, jumlahKursi, tujuan).
6	bus	Transportasi	Objek dari kelas Bus yang mewakili transportasi bus dengan

			properti tertentu (nama, jumlahKursi, tujuan).
7	Nama	String	Variabel yang diwarisi dari kelas Transportasi, menyimpan nama pesawat (misalnya, "Pesawat Air").
8	jumlahKursi	int	Variabel yang diwarisi dari kelas Transportasi, menyimpan jumlah kursi yang tersedia pada pesawat (misalnya, 120).
9	tujuan	String	Variabel yang diwarisi dari kelas Transportasi, menyimpan tujuan pesawat (misalnya, "Medan").
10	kelasLayanan	String	Parameter metode hitungHargaTiket(String kelasLayanan), menyimpan kelas layanan yang dipilih (Ekonomi, Bisnis, atau VIP).
11	basePrice	Double	Variabel lokal dalam metode hitungHargaTiket(String kelasLayanan), menyimpan harga dasar setelah override (150.000 untuk Pesawat).
12	nama	String	Menyimpan nama transportasi (misalnya, "Bus Trans").
13	jumlahKursi	int	Menyimpan jumlah kursi yang tersedia pada transportasi.
14	tujuan	String	Menyimpan tujuan transportasi (misalnya, "Bandung").
15	nama	String	Variabel yang diwarisi dari Transportasi, menyimpan nama bus (misalnya, "Bus Trans").
16	jumlahKursi	int	Variabel yang diwarisi dari Transportasi, menyimpan jumlah kursi bus

17	tujuan	String	Variabel yang diwarisi dari Transportasi, menyimpan tujuan bus.
18	kelasLayanan	String	Parameter metode hitungHargaTiket(String kelasLayanan), menyimpan kelas layanan (Ekonomi, Bisnis, atau VIP).
19	basePrice	Double	Variabel lokal dalam metode hitungHargaTiket(String kelasLayanan), menyimpan harga dasar setelah override (110.000 untuk Bus).
20	nama	String	Variabel yang diwarisi dari Transportasi, menyimpan nama kereta (misalnya, "Kereta Cepat").
21	jumlahKursi	int	Variabel yang diwarisi dari Transportasi, menyimpan jumlah kursi kereta.
22	tujuan	String	Variabel yang diwarisi dari Transportasi, menyimpan tujuan kereta.
23	kelasLayanan	String	Parameter metode hitungHargaTiket(String kelasLayanan), menyimpan kelas layanan (Ekonomi, Bisnis, atau VIP).
24	basePrice	double	Variabel lokal dalam metode hitungHargaTiket(String kelasLayanan), menyimpan harga dasar setelah override (120.000 untuk Kereta).

3. Constructor dan Method

Nama Method	Jenis Method	Fungsi

Transportasi(String nama, int jumlahKursi, String tujuan)	Constructor	Menginisialisasi objek Transportasi dengan properti nama, jumlahKursi, dan tujuan
getNama()	Method	Mengembalikan nilai nama.
setNama(String nama)	Method	Mengatur nilai nama.
getJumlahKursi()	Method	Mengembalikan nilai jumlahKursi.
setJumlahKursi(int jumlahKursi)	Method	Mengatur nilai jumlahKursi.
getTujuan()	Method	Mengembalikan nilai tujuan.
setTujuan(String tujuan)	Method	Mengatur nilai tujuan.
hitungHargaTiket()	Method	Mengembalikan harga tiket default (100.000).
@Override hitungHargaTiket()	Method	Mengembalikan harga tiket untuk Bus (100.000 + 10% = 110.000).
hitungHargaTiket(String kelasLayanan)	Method	Mengembalikan harga tiket berdasarkan kelas layanan (Ekonomi: +0%, Bisnis: +25%, VIP: +50%).
Bus(String nama, int jumlahKursi, String tujuan)	Constructor	Mengin isialisasi objek Bus dengan memanggil konstruktor Transportasi.
Kereta(String nama, int jumlahKursi, String tujuan)	Constructor	Mengin isialisasi objek Kereta dengan memanggil konstruktor Transportasi.
@Override hitungHargaTiket()	Method	Mengembalikan harga tiket untuk Kereta (100.000 + 20% = 120.000).
hitungHargaTiket(String kelasLayanan)	Method	Mengembalikan harga tiket berdasarkan kelas layanan (Ekonomi: +0%, Bisnis: +25%, VIP: +50%).
Pesawat(String nama, int jumlahKursi, String tujuan)	Cosntructor	Mengin isialisasi objek Pesawat dengan memanggil konstruktor Transportasi.
@Override hitungHargaTiket()	Method	Mengembalikan harga tiket untuk Pesawat (100.000 + 50% = 150.000).
hitungHargaTiket(String kelasLayanan)	Method	Mengembalikan harga tiket berdasarkan kelas layanan (Ekonomi: +0%, Bisnis: +25%, VIP: +50%).
main(String[] args)	Method	Metode utama untuk menjalankan program, membuat objek transportasi, dan menampilkan informasi serta harga tiket.

4. **Dokumentasi dan Pembahasan Code**

   Program ini terdiri dari empat kelas utama: Transportasi, Bus, Kereta, Pesawat, dan Main. Berikut adalah penjelasan rinci tentang implementasi kode:

   - Kelas Transportasi

     Kelas Transportasi adalah kelas induk yang menyediakan kerangka dasar untuk se mua jenis transportasi. Properti nama, jumlahKursi, dan tujuan dideklarasikan

sebagai private untuk menerapkan enkapsulasi, dengan metode getter dan setter untuk mengakses dan mengubah nilai properti. Metode hitungHargaTiket() mengembalikan harga tiket default sebesar 100.000.

```
Windsurf: Refactor | Explain
class Transportasi {
    private String nama;
    private int jumlahKursi;
    private String tujuan;

    // Constructor
    public Transportasi(String nama, int jumlahKursi, String tujuan) {
        this.nama = nama;
        this.jumlahKursi = jumlahKursi;
        this.tujuan = tujuan;
    }

    // Getters and Setters
    Windsurf: Refactor | Explain | ×
    public String getNama() {
        return nama;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setNama(String nama) {
        this.nama = nama;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public int getJumlahKursi() {
        return jumlahKursi;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setJumlahKursi(int jumlahKursi) {
        this.jumlahKursi = jumlahKursi;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public String getTujuan() {
        return tujuan;
    }

    Windsurf: Refactor | Explain | Generate Javadoc | ×
    public void setTujuan(String tujuan) {
        this.tujuan = tujuan;
    }

    // Default method for calculating ticket price
    Windsurf: Refactor | Explain | ×
    public double hitungHargaTiket() {
        return 100000.0;
    }
}
```

- Kelas Bus, Kereta, dan Pesawat

  Ketiga kelas ini adalah subclass dari Transportasi, masing-masing mengoverride metode hitungHargaTiket() untuk menghitung harga tiket sesuai aturan:

  • Bus: Menambah 10% dari harga default (100.000*1.10=110.000).

  • Kereta: Menambah 20% dari harga default (100.000 * 1.20 = 120.000). • Pesawat: Menambah 50% dari harga default (100.000 * 1.50 = 150.000). Selain itu, setiap

subclass memiliki metode overload hitungHargaTiket(String kelasLayanan) yang menyesuaikan harga berdasarkan kelas layanan: • Ekonomi: +0%. • Bisnis: +25%. • VIP: +50%. Berikut adalah contoh implementasi untuk kelas Bus:

```
Windsurf: Refactor | Explain
class Bus extends Transportasi {
    public Bus(String nama, int jumlahKursi, String tujuan) {
        super(nama, jumlahKursi, tujuan);
    }

    // Override: Add 10% to default price
    Windsurf: Refactor | Explain | ×
    @Override
    public double hitungHargaTiket() {
        return super.hitungHargaTiket() * 1.10; // 100,000 + 10% = 110,000
    }

    // Overload: Adjust price based on service class
    Windsurf: Refactor | Explain | ×
    public double hitungHargaTiket(String kelasLayanan) {
        double basePrice = hitungHargaTiket();
        if (kelasLayanan.equalsIgnoreCase(anotherString:"Ekonomi")) {
            return basePrice; // +0%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"Bisnis")) {
            return basePrice * 1.25; // +25%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"VIP")) {
            return basePrice * 1.50; // +50%
        }
        return basePrice; // Default to base price if class is invalid
    }
}
```

Berikut kelas Kereta:

```
Windsurf: Refactor | Explain
class Kereta extends Transportasi {
    public Kereta(String nama, int jumlahKursi, String tujuan) {
        super(nama, jumlahKursi, tujuan);
    }

    // Override: Add 20% to default price
    Windsurf: Refactor | Explain | ×
    @Override
    public double hitungHargaTiket() {
        return super.hitungHargaTiket() * 1.20; // 100,000 + 20% = 120,000
    }

    // Overload: Adjust price based on service class
    Windsurf: Refactor | Explain | ×
    public double hitungHargaTiket(String kelasLayanan) {
        double basePrice = hitungHargaTiket();
        if (kelasLayanan.equalsIgnoreCase(anotherString:"Ekonomi")) {
            return basePrice; // +0%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"Bisnis")) {
            return basePrice * 1.25; // +25%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"VIP")) {
            return basePrice * 1.50; // +50%
        }
        return basePrice; // Default to base price if class is invalid
    }
}
```

Berikut Kelas Pesawat:

```
Windsurf: Refactor | Explain
class Pesawat extends Transportasi {
    public Pesawat(String nama, int jumlahKursi, String tujuan) {
        super(nama, jumlahKursi, tujuan);
    }

    // Override: Add 50% to default price
    Windsurf: Refactor | Explain | ×
    @Override
    public double hitungHargaTiket() {
        return super.hitungHargaTiket() * 1.50; // 100,000 + 50% = 150,000
    }

    // Overload: Adjust price based on service class
    Windsurf: Refactor | Explain | ×
    public double hitungHargaTiket(String kelasLayanan) {
        double basePrice = hitungHargaTiket();
        if (kelasLayanan.equalsIgnoreCase(anotherString:"Ekonomi")) {
            return basePrice; // +0%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"Bisnis")) {
            return basePrice * 1.25; // +25%
        } else if (kelasLayanan.equalsIgnoreCase(anotherString:"VIP")) {
            return basePrice * 1.50; // +50%
        }
        return basePrice; // Default to base price if class is invalid
    }
}
```

- Kelas Main
  Kelas Main berfungsi untuk menjalankan program. Objek dari Bus, Kereta, dan Pesawat dibuat dan disimpan dalam array bertipe Transportasi[]. Array ini digunakan untuk menunjukkan polimorfisme, di mana metode hitungHargaTiket() dipanggil secara di namis pada setiap objek. Untuk metode overload, pemeriksaan jenis (instanceof) dan casting digunakan untuk memanggil metode yang sesuai.

```java
Windsurf: Refactor | Explain
public class Main {
    Run | Debug | Run main | Debug main | Windsurf: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        // Create objects for each transportation type
        Transportasi bus = new Bus(nama:"Bus Trans", jumlahKursi:40, tujuan:"Bandung");
        Transportasi kereta = new Kereta(nama:"Kereta Cepat", jumlahKursi:60, tujuan:"Surabaya");
        Transportasi pesawat = new Pesawat(nama:"Pesawat Air", jumlahKursi:120, tujuan:"Medan");

        // Store objects in an array to demonstrate polymorphism
        Transportasi[] transportasiArray = {bus, kereta, pesawat};

        // Loop through the array and display ticket prices
        for (Transportasi t : transportasiArray) {
            // Display default ticket price (using polymorphism)
            System.out.println(t.getNama() + " ke " + t.getTujuan() + " - Harga tiket (default): " + t.hitungHargaTiket());

            // Display ticket price with service class (using direct method call due to overloading)
            String kelasLayanan = "Ekonomi"; // Default service class
            if (t instanceof Bus) {
                kelasLayanan = "Bisnis";
                System.out.println(t.getNama() + " ke " + t.getTujuan() + " - Harga tiket (" + kelasLayanan + "): " + ((Bus)t).hitungHargaTiket(kelasLayanan));
            } else if (t instanceof Kereta) {
                kelasLayanan = "VIP";
                System.out.println(t.getNama() + " ke " + t.getTujuan() + " - Harga tiket (" + kelasLayanan + "): " + ((Kereta)t).hitungHargaTiket(kelasLayanan));
            } else if (t instanceof Pesawat) {
                kelasLayanan = "Ekonomi";
                System.out.println(t.getNama() + " ke " + t.getTujuan() + " - Harga tiket (" + kelasLayanan + "): " + ((Pesawat)t).hitungHargaTiket(kelasLayanan));
            }
        }
    }
}
```

5. Kesimpulan

Praktikum ini berhasil menunjukkan penerapan konsep pewarisan, overriding, overloading, dan polimorfisme dalam pemrograman berorientasi objek menggunakan Java. Sistem pemesanan tiket transportasi yang dibuat memungkinkan pengelolaan tiket untuk bus, kereta, dan pesawat dengan struktur kode yang terorganisir dan modular. Pewarisan memungkinkan penggunaan kembali kode dari kelas induk Transportasi, semen tara overriding dan overloading memberikan fleksibilitas dalam perhitungan harga tiket.

Polimorfisme memungkinkan penanganan objek secara seragam melalui array, sehingga mempermudah pengembangan dan perawatan sistem. Praktikum ini memberikan pemahaman mendalam tentang konsep PBO dan penerapannya dalam kasus nyata. Sekian dan terima kasih.

6. Daftar Pustaka

Modul 9: Java Inheritence (Pewarisan) 2