Penjelasan Implementasi dalam Java

Implementasi dalam Java dilakukan dengan membuat semua Kelas menjadi file berbeda, lalu terdapat main.java untuk melakukan testing terhadap implementasi yang sudah dibuat.

A. Kendaraan

```java
public abstract class Kendaraan {
    protected int nomorKendaraan;
    protected int tahunKeluaran;
    protected String merk;

    public Kendaraan() {
        this.nomorKendaraan = 0;
        this.merk = "XXX";
        this.tahunKeluaran = 0;
    }

    public Kendaraan(int nomorKendaraan, String merk, int tahunKeluaran) {
        this.nomorKendaraan = nomorKendaraan;
        this.merk = merk;
        this.tahunKeluaran = tahunKeluaran;
    }

    public Kendaraan(Kendaraan k) {
        this.nomorKendaraan = k.nomorKendaraan;
        this.merk = k.merk;
        this.tahunKeluaran = k.tahunKeluaran;
    }

    public abstract long BiayaSewa(int lamaSewa);

    public void printInfo() {
        System.out.printf("Nomor Kendaraan: %d\n", this.nomorKendaraan);
        System.out.printf("Merk: %s\n", this.merk);
        System.out.printf("Tahun Keluaran: %d\n", this.tahunKeluaran);
    }
}
```

Pertama – tama, terdapat kelas Kendaraan berupa kelas abstrak yang memiliki atribut nomorKendaraan, tahunKeluaran, serta merk. Atribut kategori dihilangkan karena dianggap sudah tidak penting karena sudah menggunakan konsep inheritance, dimana setiap kelas anak menjelaskan kategorinya, sehingga tidak perlu diberi tahu secara eksplisit kategori dari kendaraan tersebut

Kelas ini mempunyai 2 ctor (default & user-defined) , 1 cctor, dan 1 dtor. (Selain itu, juga terdapat 2 virtual method yaitu BiayaSewa untuk menghitung biaya sewa dimana implementasi diserahkan kepada kelas anak, karena tiap kategori kendaraan memiliki biaya sewa yang berbeda juga. Setelah itu ada juga PrintInfo yang berguna untuk mengeluarkan semua atribut yang dimiliki oleh kendaraan. Method PrintInfo merupakan method virtual karena implementasi diserahkan kepada anak, namun terdapat implementasi juga disini untuk atribut yang bersifat umum, sehingga anak hanya perlu memanggil fungsi ini dan menambahkan atribut uniknya masing masing.

B. Bus

```
class Bus extends Kendaraan {
    private int kapasitas;

    public Bus() {
        super();
        this.kapasitas = 0;
    }

    public Bus(int nomorKendaraan, String merk, int tahunKeluaran, int kapasitas) {
        super(nomorKendaraan, merk, tahunKeluaran);
        this.kapasitas = kapasitas;
    }

    public Bus(Bus b) {
        super(b);
        this.kapasitas = b.kapasitas;
    }

    public long BiayaSewa(int lamaSewa) {
        return 1000000 * lamaSewa;
    }

    public void printInfo() {
        super.printInfo();
        System.out.printf("Kategori: %s\n", "Bus");
        System.out.printf("Kapasitas: %d\n", this.kapasitas);
    }       You, 32 minutes ago • lupa push java …
}
```

Kelas ini merupakan turunan dari Kendaraan dan memiliki atribut tambahan yaitu kapasitas. ctor dan cctor sama seperti orang tua hanya ditambah untuk inisialisasi atribut kapasitas.

Dapat dilihat pada kode diatas terdapat implementasi biaya sewa sesuai dengan spesifikasi pada dokumen yaitu lamasewa * 1000000 dan implementasi PrintInfo ditambah dengan mengeluarkan kategori kendaraan yaitu Bus dan Atribut baru Kapasitas.

C. Mobil

```
class Mobil extends Kendaraan {
    private String supir;

    public Mobil() {
        super();
        this.supir = "XXXX";
    }

    public Mobil(int nomorKendaraan, String merk, int tahunKeluaran, String supir) {
        super(nomorKendaraan, merk, tahunKeluaran);
        this.supir = supir;
    }

    public Mobil(Mobil Mb) {
        super(Mb);
        this.supir = Mb.supir;
    }

    public long BiayaSewa(int lamaSewa) {
        return 500000 * lamaSewa;
    }

    public void printInfo() {
        super.printInfo();
        System.out.printf("Kategori: %s\n", "Mobil");
    }
}       You, 33 minutes ago • lupa push java …
```

Mobil juga merupakan anak dari kelas Kendaraan dengan atribut tambahan Supir, implementasi masih sama seperti kelas bus namun disesuaikan dengan info seharusnya dari Mobil

D. Minibus

```java
class Minibus extends Kendaraan {

    public Minibus() {
        super();
    }

    public Minibus(int nomorKendaraan, String merk, int tahunKeluaran) {
        super(nomorKendaraan, merk, tahunKeluaran);
    }

    public Minibus(Minibus M) {
        super(M);
    }

    public long BiayaSewa(int lamaSewa) {
        if (lamaSewa <= 5) {
            return 5000000;
        } else {
            return 5000000 + (500000 + (lamaSewa - 5));
        }
    }

    public void printInfo() {      You, 33 minutes ago • lupa push java …
        super.printInfo();
        System.out.printf("Kategori: %s\n", "Minibus");
    }

    public double Diskon(int lamaSewa) {
        if (lamaSewa > 10) {
            return BiayaSewa(lamaSewa) * 0.9;
        } else {
            return BiayaSewa(lamaSewa);
        }
    }
}
```

Minibus merupakan turunan dari Kendaraan dengan method tambahan yaitu Diskon. Implementasi masih sama seperti Bus dan Mobil, namun tidak ada tambahan atribut baru, sehingga hanya memanggil konstruktor dari Kendaraan. Implementasi method Diskon mengikuti spek yang diberikan.

E. Koleksi Kendaraan

```java
class KoleksiKendaraan {
    private int size;
    private int neff;
    private Kendaraan[] kumpulanKendaraan;

    public KoleksiKendaraan() {
        this.kumpulanKendaraan = new Kendaraan[100];
        this.size = 100;
        this.neff = 0;
    }

    public KoleksiKendaraan(int size) {
        this.kumpulanKendaraan = new Kendaraan[size];
        this.size = size;
        this.neff = 0;
    }

    public KoleksiKendaraan(KoleksiKendaraan k) {
        this.kumpulanKendaraan = new Kendaraan[100];
        for (int i = 0; i < k.size; i++) {
            this.kumpulanKendaraan[i] = k.kumpulanKendaraan[i];
        }
        this.size = k.size;
        this.neff = k.neff;
    }
```

```java
public void appendKendaraan(Kendaraan kendaraan)
// pengganti operator overloading << karena tidak ada op overloading di java
{
    this.kumpulanKendaraan[this.neff++] = kendaraan;
}

public void printAll() {
    for (int i = 0; i < this.neff; i++) {
        System.out.println("==================");
        this.kumpulanKendaraan[i].printInfo();
    }
}

public void appendKumpulanKendaraan(KoleksiKendaraan kk) {
    for (int i = 0; i < kk.neff; i++) {
        if (this.neff < this.size) {
            this.kumpulanKendaraan[this.neff++] = kk.kumpulanKendaraan[i];
        }
    }
}
```

Kelas koleksi kendaraan merepresentasikan kumpulan kendaraan yang dimiliki suatu tempat rental. Kelas ini sebenarnya berupa Array yang berisi objek kendaraan. Seperti biasa, terdapat 2 ctor (default dengan ukuran array 100 dan user defined dengan ukuran array tergantung masukan user), 1 cctor

Lalu, sesuai spek terdapat method printAll dengan menggunakan loop dan memanfaatkan method PrintInfo dari kelas kendaraan untuk mengeluarkan atribut yang

dimiliki oleh semua kendaraan yang ada dalam array (atau koleksi). Disini, karena pada java tidak terdapat operator overloading, yang dilakukan adalah membuat method baru dengan fungsi yang sama. yaitu appendKendaraan untuk mengganti operator << dengan parameter Kendaraan dimana method akan memasukan kendaraan baru ke array paling belakang dan appendKumpulanKendaraan untuk melakukan append semua kendaraan dari koleksi kendaraan lain dengan ada pengecekan size

F. Main

Terakhir merupakan main yang kami buat untuk mengetes program :

```
public class Main {

    Run | Debug
    public static void main(String[] args) throws Exception {
        KoleksiKendaraan KK = new KoleksiKendaraan();       You, 36 minutes ago • lupa push java …
        Kendaraan B1 = new Bus(1, "Damri", 2001, 50);
        Kendaraan M1 = new Mobil(2, "Toyota", 1999, "Dhika");
        Kendaraan MB1 = new Minibus(3, "MerkMinibus", 1700);

        KK.appendKendaraan(B1);
        KK.appendKendaraan(M1);
        KK.appendKendaraan(MB1);
        System.out.println("ISI KOLEKSI KENDARAAN 1 :");
        KK.printAll();
        System.out.println();
        System.out.println("ISI KOLEKSI KENDARAAN 2 :");
        KoleksiKendaraan KK2 = new KoleksiKendaraan();
        KK2.appendKumpulanKendaraan(KK);
        KK2.appendKendaraan(new Bus(4,"BusKeren", 2010, 100));
        KK2.appendKendaraan(new Mobil(5,"Ferrari", 2020, "Mas Rosi"));
        KK2.printAll();
    }

}
```

Berikut output :

```
ISI KOLEKSI KENDARAAN 1 :
==================
Nomor Kendaraan: 1
Merk: Damri
Tahun Keluaran: 2001
Kategori: Bus
Kapasitas: 50
==================
Nomor Kendaraan: 2
Merk: Toyota
Tahun Keluaran: 1999
Kategori: Mobil
==================
Nomor Kendaraan: 3
Merk: MerkMinibus
Tahun Keluaran: 1700
Kategori: Minibus

ISI KOLEKSI KENDARAAN 2 :
==================
Nomor Kendaraan: 1
Merk: Damri
Tahun Keluaran: 2001
Kategori: Bus
Kapasitas: 50
==================
Nomor Kendaraan: 2
Merk: Toyota
Tahun Keluaran: 1999
Kategori: Mobil
==================
Nomor Kendaraan: 3
Merk: MerkMinibus
Tahun Keluaran: 1700
Kategori: Minibus
==================
Nomor Kendaraan: 4
Merk: BusKeren
Tahun Keluaran: 2010
Kategori: Bus
Kapasitas: 100
==================
Nomor Kendaraan: 5
Merk: Ferrari
Tahun Keluaran: 2020
Kategori: Mobil
```

Main yang kami buat disini sama dengan main yang ada pada implementasi cpp, dan dapat dilihat output yang didapatkan sama.