Penjelasan Implementasi dalam CPP

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

A. Kendaraan

```cpp
#include "Kendaraan.hpp"

Kendaraan::Kendaraan()
{
    this->nomorKendaraan = 0;
    this->merk = "XXX";
    this->tahunKeluaran = 0;
}

Kendaraan::Kendaraan(int nomorKendaraan, string merk, int tahunKeluaran)
{
    this->nomorKendaraan = nomorKendaraan;
    this->merk = merk;
    this->tahunKeluaran = tahunKeluaran;
}

Kendaraan::Kendaraan(const Kendaraan &k)
{
    this->nomorKendaraan = k.nomorKendaraan;
    this->merk = k.merk;
    this->tahunKeluaran = k.tahunKeluaran;
}

Kendaraan::~Kendaraan()
{ /*tidak butuh*/
}

int Kendaraan :: BiayaSewa(int lamaSewa)
{return 0;}       You, 11 minutes ago • selese nih …

void Kendaraan::PrintInfo()
{
    cout << "Nomor Kendaraan: " << this->nomorKendaraan << endl;
    cout << "Merk: " << this->merk << endl;
    cout << "Tahun Keluaran: " << this->tahunKeluaran << endl;
}
```

Pertama – tama, terdapat kelas Kendaraan yaitu 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 (namun dtor digunakan karena tidak ada alokasi memori heap). 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

```cpp
#include "Bus.hpp"

// ctor
Bus ::Bus() : Kendaraan()
{
    this->kapasitas = 0;
}
// user-defined ctor
Bus ::Bus(int kapasitas, int nomorKendaraan, string merk, int tahunKeluaran) : Kendaraan(nomorKendaraan, merk, tahunKeluaran)
{
    this->kapasitas = kapasitas;
}

// cctor
Bus ::Bus(const Bus &b) : Kendaraan(b)
{
    this->kapasitas = b.kapasitas;      You, yesterday • versi cpp …
}

// dtor: tidak diperlukan karena mengalokasikan memori heap
Bus ::~Bus(){};

int Bus ::BiayaSewa(int lamaSewa)
{
    return 1000000 * lamaSewa;
}

void Bus ::PrintInfo()
{
    Kendaraan::PrintInfo();
    cout << "Kategori: Bus" << endl;
    cout << "Kapasitas: " << this->kapasitas << endl;
}
```

Kelas ini merupakan turunan dari Kendaraan dan memiliki atribut tambahan yaitu kapasitas. ctor, cctor dan dtor 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

```cpp
#include "Mobil.hpp"      You, yesterday • versi cpp …

Mobil ::Mobil() : Kendaraan()
{
    this->supir = "XXXX";
}
// user-defined ctor
Mobil ::Mobil(string supir, int nomorKendaraan, string merk, int tahunKeluaran) : Kendaraan(nomorKendaraan, merk, tahunKeluaran)
{
    this->supir = supir;
}
// cctor
Mobil ::Mobil(const Mobil &m) : Kendaraan(m)
{
    this->supir = m.supir;
}
// dtor: tidak diperlukan karena mengalokasikan memori heap
Mobil ::~Mobil(){};

void Mobil ::PrintInfo()
{
    Kendaraan::PrintInfo();
    cout << "Kategori: Mobil" << endl;
    cout << "Supir: " << this->supir << endl;
}

int Mobil ::BiayaSewa(int lamaSewa)
{
    return 500000 * lamaSewa;
}
```

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

```cpp
#include "Minibus.hpp"      You, yesterday • versi cpp …

Minibus::Minibus() : Kendaraan() {}

// user-defined ctor
Minibus::Minibus(int nomorKendaraan, string merk, int tahunKeluaran) : Kendaraan(nomorKendaraan, merk, tahunKeluaran) {}

// cctor
Minibus::Minibus(const Minibus &m) : Kendaraan(m) {}

// dtor: tidak diperlukan karena mengalokasikan memori heap
Minibus::~Minibus() {}

void Minibus ::printInfo()
{
    Kendaraan::PrintInfo();
    cout << "Kategori: Minibus" << endl;
}

int Minibus ::BiayaSewa(int lamaSewa)
{
    if (lamaSewa >= 5)
    {
        return 5000000;
    }
    else
    {
        return 5000000 + (500000 * (lamaSewa - 5));
    }
}

double Minibus ::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.

E. Koleksi Kendaraan

```cpp
#include "KoleksiKendaraan.hpp"

KoleksiKendaraan::KoleksiKendaraan()
{
    this->kumpulanKendaraan = new Kendaraan[100];
    this->size = 100;
    this->neff = 0;
}

KoleksiKendaraan::KoleksiKendaraan(int size)
{
    this->kumpulanKendaraan = new Kendaraan[size];
    for (int i = 0; i < size; i++)
    {
        this->kumpulanKendaraan[i] = Kendaraan();
    }
    this->size = size;
    this->neff = 0;
}

KoleksiKendaraan::KoleksiKendaraan(const KoleksiKendaraan &
{
    this->kumpulanKendaraan = new Kendaraan[k.size];
    for (int i = 0; i < k.neff; i++)
    {
        this->kumpulanKendaraan[i] = k.kumpulanKendaraan[i]
    }
    this->size = k.size;
    this->neff = k.neff;
}
      You, yesterday • versi cpp …
```

```cpp
KoleksiKendaraan::~KoleksiKendaraan()
{
    delete[] this->kumpulanKendaraan;
}

void KoleksiKendaraan::PrintAll()
{
    for (int i = 0; i < this->neff; i++)
    {
        this->kumpulanKendaraan[i].PrintInfo();
    }
}

void KoleksiKendaraan::operator<<(Kendaraan &k)
{
    this->kumpulanKendaraan[neff] = k;
    this->neff += 1;
}

void KoleksiKendaraan::operator<<(KoleksiKendaraan &k)
{
    for (int i = 0; i < k.neff; i++)
    {
        if(this->neff < this->size)
        {
            (*this) << k.kumpulanKendaraan[i];      You, now
        }
    }
}
```

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, 1 dtor (disini dtor diimplementasikan karena terdapat array yang dialokasikan).

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) dan memiliki 2 operator overloading untuk "<<". Terdapat 2 yaitu parameter Kendaraan dan KoleksiKendaraan. Pada parameter Kendaraan intinya akan memasukkan kendaraan baru di paling belakang dan pada parameter KoleksiKendaraan akan memasukkan semua kendaraan dari koleksi kendaraan parameter juga dicek agar tidak melebihi kapasitas yang ada

F. Main

Terakhir merupakan main yang kami buat untuk mengetes program :

```cpp
#include "KoleksiKendaraan/KoleksiKendaraan.hpp"
#include "Kendaraan/Kendaraan.hpp"
#include "Kendaraan/Mobil.hpp"
#include "Kendaraan/Minibus.hpp"
#include "Kendaraan/Bus.hpp"

int main()
{
    KoleksiKendaraan *KK = new KoleksiKendaraan();      You, 29 minutes a
    Kendaraan *B1 = new Bus(50,1,"Damri", 2001);
    Kendaraan *M1 = new Mobil("Dhika",2, "Toyota", 1999);
    Kendaraan *MB1 = new Minibus(3, "MerkMinibus", 1700);

    *KK << *B1;
    *KK << *M1;
    *KK << *MB1 ;
    cout << "ISI KOLEKSI KENDARAAN 1 :" << endl;
    KK->PrintAll();
    cout << endl;
    cout << "ISI KOLEKSI KENDARAAN 2 :" << endl;
    KoleksiKendaraan *KK2 = new KoleksiKendaraan();
    *KK2<<*KK;
    *KK2 << *new Bus(100,4,"BusKeren",2020);
    *KK2 << *new Mobil("Mas Rosi", 5, "Ferrari", 2020);
    KK2->PrintAll();
}
```

Berikut output :

```
ISI KOLEKSI KENDARAAN 1 :
Nomor Kendaraan: 1
Merk: Damri
Tahun Keluaran: 2001
Nomor Kendaraan: 2
Merk: Toyota
Tahun Keluaran: 1999
Nomor Kendaraan: 3
Merk: MerkMinibus
Tahun Keluaran: 1700

ISI KOLEKSI KENDARAAN 2 :
Nomor Kendaraan: 1
Merk: Damri
Tahun Keluaran: 2001
Nomor Kendaraan: 2
Merk: Toyota
Tahun Keluaran: 1999
Nomor Kendaraan: 3
Merk: MerkMinibus
Tahun Keluaran: 1700
Nomor Kendaraan: 4
Merk: BusKeren
Tahun Keluaran: 2020
Nomor Kendaraan: 5
Merk: Ferrari
Tahun Keluaran: 2020
```