

Modul 7

1. Capaian Pembelajaran Matakuliah (CPMK)

CPMK142:

Mampu menerapkan teknik pemrograman terstruktur, modular, dan berorientasi objek untuk menyelesaikan permasalahan komputasi sesuai kebutuhan aplikasi sistem komputer.

Indikator Ketercapaian:

Mahasiswa mampu memahami abstraction.

2. Dasar Teori

1. Apa itu Abstraksi (Abstraction)?

Abstraksi adalah pilar utama OOP yang berfokus pada **apa yang dilakukan oleh suatu objek** daripada **bagaimana objek tersebut melakukannya**. Abstraksi menyembunyikan detail kompleksitas internal/implementasi dan hanya menyajikan antarmuka esensial yang relevan kepada pengguna.

2. Pure Virtual Function dan Abstract Class

Dalam C++, kelas abstrak (*abstract class*) adalah kelas yang **tidak dapat diinstansiasikan langsung menjadi objek**. Kelas ini dirancang khusus untuk menjadi kelas induk (*base class*) yang menetapkan kerangka kerja atau kontrak bagi kelas-kelas turunannya.

Sebuah kelas di C++ otomatis menjadi *abstract class* jika memiliki setidaknya satu **Pure Virtual Function**

```
class AbstractClass {
public:
    // Pure Virtual Function (ditandai dengan = 0)
    virtual void fungsiAbstrak() = 0;
};
```

Pure Virtual Function: Fungsi virtual yang tidak memiliki definisi/implementasi pada kelas induk (diakhiri dengan = 0). Kelas anak **wajib** meng-overriding dan mengimplementasikan fungsi ini

3. Interface dalam C++

Bahasa C++ tidak memiliki kata kunci interface khusus seperti pada Java atau C#. Namun, konsep **Interface** di C++ diimplementasikan dengan membuat sebuah

Abstract Class yang seluruh metodenya adalah Pure Virtual Function dan tidak memiliki variabel/atribut data (selain konstanta).

Interface menentukan kumpulan standar metode yang harus dimiliki oleh kelas mana pun yang mengimplementasikannya, tanpa memedulikan detail struktur internal kelas tersebut

3. Alat dan Bahan

Hardware :

1. Laptop/PC
2. LCD/proyektor
3. Jaringan internet

Software :

1. Sistem operasi Windows/Linux/macOS
2. Java Development Kit (JDK)
3. IDE pemrograman: NetBeans / IntelliJ IDEA / Visual Studio Code
4. LMS untuk materi dan aktivitas pembelajaran

4. Langkah Kerja / Prosedur Praktikum

Buat berkas baru bernama modul7_abstraction.cpp.

Menuliskan Interface, Abstract Class, dan Derived Class:

Langkah 1.

Ketikkan kode berikut di dalam berkas modul7_abstraction.cpp:

```
#include <iostream>
#include <string>

// 1. PURE INTERFACE (Hanya berisi Pure Virtual Function)
class Pembayaran {
public:
    // Pure Virtual Functions
    virtual void prosesPembayaran(double jumlah) = 0;
    virtual void cetakResi() = 0;

    // Virtual Destructor
    virtual ~Pembayaran() {}
};

// 2. IMPLEMENTASI INTERFACE PADA CLASS CONCRETE 1
class KartuKredit : public Pembayaran {
```

```

private:
    std::string nomorKartu;

public:
    KartuKredit(std::string no) : nomorKartu(no) {}

    // Wajib mengimplementasikan seluruh Pure Virtual Function
    void prosesPembayaran(double jumlah) override {
        std::cout << "[Kartu Kredit] Memproses pembayaran sebesar Rp " << jumlah
            << " menggunakan Kartu: " << nomorKartu << std::endl;
    }

    void cetakResi() override {
        std::cout << "[Resi] Transaksi Kartu Kredit Berhasil." << std::endl;
    }
};

// 3. IMPLEMENTASI INTERFACE PADA CLASS CONCRETE 2
class EWallet : public Pembayaran {
private:
    std::string nomorHP;

public:
    EWallet(std::string hp) : nomorHP(hp) {}

    // Wajib mengimplementasikan seluruh Pure Virtual Function
    void prosesPembayaran(double jumlah) override {
        std::cout << "[E-Wallet] Memotong saldo sebesar Rp " << jumlah
            << " dari akun E-Wallet: " << nomorHP << std::endl;
    }

    void cetakResi() override {
        std::cout << "[Resi] Transaksi E-Wallet Berhasil Diproses." << std::endl;
    }
};

```

Lanjutkan dengan menambahkan fungsi main() di bawah deklarasi kelas pada berkas yang sama:

```

int main() {
    // COBA UN-COMMENT BARIS DI BAWAH INI UNTUK MELIHAT ERROR:
    // Pembayaran p; // Error: cannot declare variable 'p' to be of abstract type
    'Pembayaran'

    std::cout << "=== TRANSAKSI 1 (KARTU KREDIT) ===" << std::endl;
    // Menggunakan Pointer Interface/Abstract Class untuk menampung objek konkrit
    Pembayaran* metodel = new KartuKredit("4567-8901-2345");
    metodel->prosesPembayaran(750000.0);
    metodel->cetakResi();

    std::cout << "\n=== TRANSAKSI 2 (E-WALLET) ===" << std::endl;
    Pembayaran* metode2 = new EWallet("0812-3456-7890");
    metode2->prosesPembayaran(150000.0);
    metode2->cetakResi();

    // Pembersihan Memori
    delete metodel;
    delete metode2;

    return 0;
}

```

Buka Terminal di VS Code.

Jalankan perintah kompilasi. Amati dan analisis program.

5. Lembar Kerja

Latihan 7

Buatlah program C++ lengkap dengan spesifikasi berikut:

- Buat **Interface / Abstract Class** bernama *Notifikasi* yang memiliki dua *pure virtual function*:

1. `kirimPesan(std::string pesan) = 0`
2. `getStatus() = 0`

- Buat **Derived Class 1** bernama *EmailNotifikasi* yang mengimplementasikan kedua metode tersebut untuk mensimulasikan pengiriman surel.
- Buat **Derived Class 2** bernama *SMSNotifikasi* yang mengimplementasikan kedua metode tersebut untuk mensimulasikan pengiriman pesan SMS.
- Tuliskan fungsi `main()` untuk menguji pengiriman pesan melalui pointer tipe *Notifikasi** untuk kedua kelas turunan tersebut

HASIL PERCOBAAN:

**Screenshot Hasil LATIHAN 7 dan Berikan Analisis.
(UPLOAD ke LMS → TUGAS PRAKTIKUM 7)**

Tugas Mandiri 7

(Kerjakan Tugas Mandiri 7 Upload ke LMS → **TUGAS TEORI 7**)

1. Jelaskan perbedaan mendasar antara *Encapsulation* (Enkapsulasi) dan *Abstraction* (Abstraksi) dalam konteks pemrograman berorientasi objek?
2. Bagaimana cara kita merealisasikan konsep *Interface* dalam bahasa C++ yang secara native tidak memiliki kata kunci interface? Apa syarat agar suatu *Abstract Class* dapat dikategorikan sebagai *Pure Interface*?
3. Mengapa sebuah *Abstract Class* tidak bisa diinstansiasikan langsung menggunakan kata kunci *new* atau pembuatan objek secara langsung? Apa konsekuensi yang terjadi jika kelas anak tidak meng-overriding *Pure Virtual Function* yang diwariskan dari kelas induk?



Program Studi
SISTEM KOMPUTER