

Modul 6

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 menerapkan polymorphism.

2. Dasar Teori

1. Apa itu Polymorphism (Polimorfisme)?

Polimorfisme berasal dari bahasa Yunani yang berarti "banyak bentuk". Dalam OOP, polimorfisme adalah kemampuan suatu metode atau antarmuka (*interface*) untuk memiliki perilaku yang berbeda-beda tergantung dari objek yang memanggilya.

Polimorfisme terbagi menjadi dua jenis utama:

1. **Compile-time Polymorphism (Static Binding):** Diimplementasikan melalui *Method Overloading* dan *Operator Overloading* (telah dipelajari di Modul 4). Keputusan pemanggilan fungsi ditentukan saat kompilasi.
2. **Runtime Polymorphism (Dynamic Binding):** Diimplementasikan melalui *Method Overriding* dan *Virtual Function*. Keputusan pemanggilan fungsi ditunda hingga program berjalan (*runtime*).

2. Virtual Function dan Method Overriding

- **Method Overriding:** Terjadi ketika kelas anak (*derived class*) menulis ulang implementasi metode yang sudah ada pada kelas induk (*base class*).
- **Virtual Function:** Metode pada kelas induk yang dideklarasikan dengan kata kunci **virtual**. Dideklarasikan agar kompilator tahu bahwa metode ini dapat di-overriding oleh kelas anak dan dipanggil secara dinamis saat *runtime*.

3. Dynamic Binding (Late Binding)

Dynamic Binding terjadi ketika pemanggilan fungsi dihubungkan dengan kode eksekusi pada saat program dijalankan, bukan saat dikompilasi. Dalam C++, hal ini dicapai dengan menggabungkan **kata kunci virtual** dan pemanggilan metode melalui **pointer/referensi kelas induk** yang menunjuk ke objek kelas anak

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 modul6_polymorphism.cpp.

Menuliskan Class Induk dengan Virtual Function dan Class Anak:

Langkah 1.

Ketikkan kode berikut di dalam berkas modul6_polymorphism.cpp:

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

// 1. BASE CLASS
class BangunDatar {
public:
    // Virtual Function (Memungkinkan Dynamic Binding)
    virtual void gambar() {
        std::cout << "Menggambar bangun datar umum." << std::endl;
    }

    virtual double hitungLuas() {
        return 0.0;
    }

    // Virtual Destructor untuk memastikan pembersihan memori aman
    virtual ~BangunDatar() {}
};

// 2. DERIVED CLASS 1
```

```

class Lingkaran : public BangunDatar {
private:
    double jariJari;

public:
    Lingkaran(double r) {
        jariJari = r;
    }

    // Method Overriding
    void gambar() override {
        std::cout << "Menggambar Lingkaran 0" << std::endl;
    }

    double hitungLuas() override {
        return 3.14159 * jariJari * jariJari;
    }
};

// 3. DERIVED CLASS 2
class Segitiga : public BangunDatar {
private:
    double alas;
    double tinggi;

public:
    Segitiga(double a, double t) {
        alas = a;
        tinggi = t;
    }

    // Method Overriding
    void gambar() override {
        std::cout << "Menggambar Segitiga /\\" << std::endl;
    }

    double hitungLuas() override {
        return 0.5 * alas * tinggi;
    }
};

```

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

```
int main() {
    // Menggunakan Pointer Kelas Induk untuk menunjuk ke Objek Kelas Anak
    BangunDatar* bentuk1 = new Lingkaran(7.0);
    BangunDatar* bentuk2 = new Segitiga(10.0, 5.0);

    std::cout << "=== PENGUJIAN DYNAMIC BINDING (SINGLE POINTER) ===" << std::endl;
    // Pemanggilan method ditentukan saat RUNTIME, bukan compile-time
    bentuk1->gambar();
    std::cout << "Luas Lingkaran: " << bentuk1->hitungLuas() << std::endl;

    std::cout << std::endl;

    bentuk2->gambar();
    std::cout << "Luas Segitiga : " << bentuk2->hitungLuas() << std::endl;

    std::cout << "\n=== PENGUJIAN POLIMORFISME DALAM ARRAY/COLLECTION ===" <<
    std::endl;
    // Menyimpan berbagai macam objek turunan dalam satu wadah/array bertipe pointer
    Induk
    std::vector<BangunDatar*> daftarBentuk;
    daftarBentuk.push_back(new Lingkaran(10.0));
    daftarBentuk.push_back(new Segitiga(6.0, 8.0));
    daftarBentuk.push_back(new Lingkaran(3.5));

    for (size_t i = 0; i < daftarBentuk.size(); i++) {
        std::cout << "Bentuk ke-" << (i + 1) << ":" << std::endl;
        daftarBentuk[i]->gambar();
        std::cout << "Luas: " << daftarBentuk[i]->hitungLuas() << std::endl;
        std::cout << "-----" << std::endl;
    }

    // Pembersihan Memori (Dealokasi)
    delete bentuk1;
    delete bentuk2;
    for (auto b : daftarBentuk) {
        delete b;
    }
}
```

```

    }

    return 0;
}

```

Buka Terminal di VS Code.

Jalankan perintah kompilasi. Amati dan analisis program.

5. Lembar Kerja

Latihan 6

Buatlah program C++ lengkap dengan spesifikasi berikut:

- Buat **Base Class** Hewan yang memiliki *virtual function* bersuara().
- Buat **Derived Class 1** Kucing yang meng-*overriding* bersuara() dengan menampilkan pesan "Meow meow!".
- Buat **Derived Class 2** Anjing yang meng-*overriding* bersuara() dengan menampilkan pesan "Guk guk!".
- Di dalam fungsi main(), buatlah sebuah *array* atau *vector* bertipe pointer Hewan* yang menampung objek Kucing dan Anjing.
- Gunakan perulangan (*looping*) untuk memanggil metode bersuara() pada seluruh objek dalam wadah tersebut sehingga membuktikan penerapan polimorfisme secara dinamis!

HASIL PERCOBAAN:

Screenshot Hasil LATIHAN 6 dan Berikan Analisis.
(UPLOAD ke LMS → TUGAS PRAKTIKUM 6)

Tugas Mandiri 6

(Kerjakan Tugas Mandiri 6 Upload ke LMS → **TUGAS TEORI 6**)

1. Jelaskan perbedaan antara Static Binding (Early Binding) dan Dynamic Binding (Late Binding)! Berikan contoh situasi pemrograman kapan kita harus memanfaatkan Dynamic Binding?
2. Apa yang terjadi jika kata kunci virtual pada deklarasi fungsi gambar() di kelas BangunDatar dihilangkan, kemudian kita memanggil bentuk1->gambar() di mana bentuk1 adalah pointer tipe BangunDatar* yang menunjuk ke objek Lingkaran? Jelaskan analisis Anda?
3. Mengapa penggunaan penanda override pada method kelas anak sangat disarankan dalam C++ modern? Jelaskan bahaya atau kesalahan yang bisa dicegah oleh kata kunci tersebut



Program Studi
SISTEM KOMPUTER