MODUL 6 & 7 – DEEP DIVE

Inheritance, Polimorfisme & Objek Dinamis

Analisis Mendalam PBO & Studi Kasus JavaScript

Eksplorasi komprehensif dari Class Diagram, Visibility Modifiers, Constructor Chaining, Overloading, Overriding, Upcasting, Downcasting, hingga 5 Studi Kasus Lengkap.

PETA MODUL LENGKAP

Agenda Pembelajaran Mendalam

1. Class Diagram & Views

Logical view, notasi class, atribut, operasi, dan visibility (-/#/+).

2. Inheritance & Super

Keyword extends, hirarki kelas, keyword super, dan constructor chaining.

3. Overloading & Overriding

Static vs Dynamic polymorphism, aturan parameter, dan return type.

4. Upcasting & Downcasting

Konversi referensi objek, penanganan dinamis, dan studi kasus JS.

UML ARCHITECTURE

Logical View dalam Sistem Berorientasi Objek

Jantung OOP (Class)

Sistem terdiri dari kumpulan objek. Class menjelaskan jenis objek yang dimiliki sistem, sedangkan class diagram menunjukkan relasi antar class tersebut.

Hubungan Use Case & Class

Use case menggambarkan perilaku fungsional sistem, sementara class diagram merinci entitas struktural yang diperlukan untuk memenuhi use case tersebut.



Statis & Struktural

Class diagram bersifat statis; menggambarkan relasi antar kelas secara arsitektural selama tahap desain dan analisis sistem.

UML COMPONENTS

Anatomi & Tiga Area Pokok Class

1. Bagian Atas (Nama)

Berisi nama class (PascalCase, kata benda tunggal) serta opsional path name package.

2. Bagian Tengah (Atribut)

Merepresentasikan state/properti objek. Ditulis dengan huruf awal kecil (camelCase) beserta tipe datanya.

3. Bagian Bawah (Operasi)

Method atau fungsi yang dapat dilakukan objek untuk memanipulasi keadaan data didalamnya.

VISUAL UML REPRESENTATION

Sample Class Diagram: Customer, Person & Toolbar

Struktur Kompleks Kelas (UML)

Berdasarkan materi modul, diagram kelas memperlihatkan relasi antar entitas seperti **Customer**, **Person**, dan **MemberCustomer** lengkap dengan atribut private (-) dan operasi public (+).



ENCAPSULATION CONTROL

Analisis Mendalam Kontrol Akses (Visibility)

- Private (-)

Hanya dapat dipanggil dari dalam class itu sendiri. Melindungi data dari modifikasi langsung di luar class.

Protected (#)

Dapat diakses oleh class bersangkutan serta anak-anak kelas yang mewarisinya (subclass dalam hirarki).

+ Public (+)

Dapat dipanggil oleh siapa saja tanpa batasan package maupun hirarki subclass.

PEWARISAN SIFAT

Hubungan Superclass & Subclass

Definisi Pewarisan

Hubungan antara kelas yang lebih umum (superclass) dan kelas yang lebih khusus (subclass). Subclass secara otomatis mewarisi seluruh atribut dan perilaku valid dari superclass.

Keyword Extends

Digunakan dalam sintaks pemrograman untuk mendeklarasikan hubungan pewarisan antar kelas.

```
// Sintaks Pewarisan Dasar
class Employee {
  constructor(name) { this.name = name; }
}
class Manager extends Employee {
  // Mewarisi name dan menambah departemen
  constructor(name, dept) {
    super(name);
    this.department = dept;
  }
}
```

PARENT REFERENCE

Peran Keyword Super pada Konstruktor

Konstruktor Tidak Diwariskan

Meskipun properti dan method diwariskan, konstruktor superclass tidak diwariskan secara langsung ke subclass.

Aturan Wajib `super()`

Panggilan konstruktor parent menggunakan `super(parameter)` harus ditempatkan pada baris pertama di dalam konstruktor subclass.

```
// Constructor Chaining

class Binatang {
  constructor(umur) { this.umur = umur; }
}
class Herbivora extends Binatang {
  constructor(umur, jenis) {
    super(umur); // Mengirim parameter ke parent
    this.jenis = jenis;
  }
}
```

POLYMORPHISM CORE

Konsep Polimorfisme Objek Dinamis

Banyak Bentuk (Polymorphism)

Kemampuan reference variabel untuk mengubah perilaku menurut objek yang dijadikan acuannya. Memungkinkan subkelas mengeksekusi perintah berbeda pada method warisan superkelas.

1 **Static Polymorphism**
Dikenal sebagai **Method Overloading** (kompilasi).

2 **Dynamic Polymorphism**
Dikenal sebagai **Method Overriding** (eksekusi).

STATIC POLYMORPHISM

Aturan & Mekanisme Method Overloading

Jumlah Parameter

Metoda bernama sama menerima jumlah argumen berbeda (misal: 2 parameter vs 3 parameter).

Tipe Data Parameter

Variasi tipe data parameter input (misal: integer, float, string, atau double).

Urutan Tipe Data

Urutan parameter yang berbeda (misal: `(int, double)` dibandingkan dengan `(double, int)`).

JAVASCRIPT SIMULATION

Implementasi Method Overloading di JavaScript

Simulasi Operator Overloading

Karena JavaScript tidak mendukung overloading native seperti Java, kita memanfaatkan rest parameter (`...args`) untuk menangani berbagai variasi input.

```
class Kalkulator {
  hitung( ...args) {
    if (args.length === 2) {
      console.log(`Jumlah 2 angka = ${args[0] + args[1]}`);
    } else if (args.length === 3) {
      console.log(`Jumlah 3 angka = ${args[0] + args[1] + args[2]}`);
    } else {
      console.log(`Input tidak dikenali`);
    }
  }
}
const calc = new Kalkulator();
calc.hitung(10, 20);        // 2 angka
calc.hitung(5, 10, 15);     // 3 angka
```

DYNAMIC POLYMORPHISM

Aturan & Ketentuan Method Overriding

Definisi Overriding

Method di superclass yang ditulis ulang (dideklarasikan kembali) oleh subclass dengan implementasi perintah yang berbeda sesuai kebutuhan spesifik kelas anak.

Ketentuan Utama

Nama method harus sama, jumlah/tipe parameter harus sama, serta hak akses tidak boleh lebih ketat dibandingkan parent-nya.

```
// Overriding di JavaScript
class Kendaraan {
  bergerak() { return "Kendaraan bergerak"; }
}
class Mobil extends Kendaraan {
  bergerak() { return "Mobil bergerak di jalan raya"; }
}
```

RESOLUTION ORDER

Prioritas Eksekusi Data & Method

Prioritas Kelas Sendiri

Ketika objek memanggil atribut atau method dengan nama yang sama, sistem akan mengeksekusi di kelasnya sendiri terlebih dahulu sebelum mengecek ke parent.

Penggunaan Kata Kunci Super

Jika ingin memaksa mengakses data atau method dari parent/superclass yang tertimpa, gunakan kata kunci `super`.

```
class Binatang {
  getUmur() { return 12; }
}
class Herbivora extends Binatang {
  getUmur() { return 25; } // Override
  testParent() {
    return super.getUmur(); // Mengakses parent
  }
}
const h = new Herbivora();
console.log(h.getUmur());     // 25 (milik sendiri)
console.log(h.testParent()); // 12 (milik parent)
```

OBJECT CASTING (1)

Upcasting Objek dalam PBO

Definisi Upcasting

Mengubah referensi objek dari subkelas (subclass) menjadi tipe superkelas (superclass). Bersifat aman dan otomatis karena setiap instans anak adalah bagian dari induk.

Contoh Konseptual

Misalnya membuat objek `Student` atau `Employee` namun diacu oleh variabel bertipe `Person`.

```
// Ilustrasi Upcasting
class Person {}
class Student extends Person {}

// Person p menunjuk ke instance Student (Upcasting)
let p = new Student();
```

OBJECT CASTING (2)

Downcasting Objek dalam PBO

Definisi Downcasting

Mengembalikan referensi superclass kembali ke tipe subclass aslinya. Diperlukan ketika kita ingin mengakses method atau atribut unik yang hanya ada di subclass.

Keleluasaan JavaScript

Dalam JavaScript, karena tipenya dinamis, referensi objek secara alami mendukung akses tanpa cast eksplisit yang kaku.

```
// Ilustrasi Downcasting

let person = new Student(); // Upcast
// Downcasting ke spesifik Student
let studentRef = person;
studentRef.tampilkanInfoMahasiswa();
```

STUDI KASUS 1

Hierarki Manusia, Dosen, & Mahasiswa

Skenario Sistem Akademik

Kelas induk `Manusia` dengan method `siapaKamu()` di-override oleh kelas `Dosen` dan `Mahasiswa` untuk menampilkan identitas spesifik.

```
class Manusia {
  siapaKamu() { return "Saya Manusia"; }
}
class Dosen extends Manusia {
  siapaKamu() { return "Saya Dosen"; }
}
class Mahasiswa extends Manusia {
  siapaKamu() { return "Saya Mahasiswa"; }
}
const wargaKampus = [new Dosen(), new Mahasiswa()];
wargaKampus.forEach(w ⇒ console.log(w.siapaKamu()));
```

STUDI KASUS 2

Polimorfisme Pergerakan Binatang & Mamalia

Skenario Sistem Biologi

Kelas `Binatang` memiliki method `bergerak()` dan `berkembangBiak()`. Kelas `Mamalia` melakukan overriding pada method `bergerak()`.

```
class Binatang {
  bergerak() { return "Binatang bergerak sesuai kemampuannya"; }
  berkembangBiak() { return "Binatang berkembang biak"; }
}
class Mamalia extends Binatang {
  bergerak() { return "Mamalia bergerak sebagian besar dengan kaki"; }
  berlari() { return "Sebagian besar mamalia dapat berlari"; }
}
const m = new Mamalia();
console.log(m.bergerak());
console.log(m.berkembangBiak());
```

STUDI KASUS 3

Perhitungan Gaji Polimorfik Karyawan

Skenario Manajemen Gaji

Menghitung total kompensasi bulanan untuk berbagai kategori pegawai melalui pemanggilan method `getGaji()` secara seragam.

```
class Pegawai {
  getGaji() { return 0; }
}
class Tetap extends Pegawai {
  getGaji() { return 6000000; }
}
class ParuhWaktu extends Pegawai {
  getGaji() { return 2500000; }
}
const daftarPegawai = [new Tetap(), new ParuhWaktu()];
daftarPegawai.forEach(p ⇒ console.log(`Gaji: Rp${p.getGaji()}`));
```

STUDI KASUS 4

Diskon Produk Berbasis Kategori

Skenario Toko Online

Kelas `Produk` dengan method `hitungHarga()` di-override oleh `Buku` dan `Elektronik` untuk memberikan persentase potongan harga yang berbeda.

```
class Produk {
  constructor(harga) { this.harga = harga; }
  hitungHarga() { return this.harga; }
}
class Buku extends Produk {
  hitungHarga() { return this.harga * 0.85; } // Diskon 15%
}
class Elektronik extends Produk {
  hitungHarga() { return this.harga * 0.90; } // Diskon 10%
}
const keranjang = [new Buku(100000), new Elektronik(2000000)];
keranjang.forEach(item ⇒ console.log(`Harga Akhir: Rp${item.hitungHarga()}`));
```

STUDI KASUS 5

Gateway Pembayaran Multi-Channel

Skenario Checkout

Memproses transaksi pembayaran melalui berbagai kanal (Virtual Account, E-Wallet, Kartu Kredit) menggunakan method polimorfik `bayar()`.

```
class Pembayaran {
  bayar(jumlah) { return `Bayar Rp${jumlah}`; }
}
class VirtualAccount extends Pembayaran {
  bayar(jumlah) { return `Transfer VA sejumlah Rp${jumlah}`; }
}
class DompetDigital extends Pembayaran {
  bayar(jumlah) { return `Potong saldo e-wallet Rp${jumlah}`; }
}
function prosesCheckout(channel, nominal) {
  console.log(channel.bayar(nominal));
}
prosesCheckout(new VirtualAccount(), 500000);
prosesCheckout(new DompetDigital(), 75000);
```

ARCHITECTURAL BENEFITS

Keunggulan Penerapan Polimorfisme

Fleksibilitas Tinggi

Memungkinkan penambahan objek atau fitur baru tanpa mengubah kode pemanggil utama.

Reduksi Kompleksitas

Menghilangkan penggunaan struktur percabangan `switch-case` atau `if-else` yang kompleks.

Maintainability

Kode menjadi lebih bersih, terstruktur rapi, dan mudah dikembangkan dalam skala besar.

SUMMARY MATRIX

Matriks Perbandingan Overloading & Overriding

Method Overloading

- Terjadi dalam satu kelas yang sama.
- Parameter wajib berbeda (jumlah/tipe).
- Dikenal sebagai Static Polymorphism.
- Return type boleh berbeda.

Method Overriding

- Terjadi antara superclass dan subclass.
- Parameter dan nama wajib sama persis.
- Dikenal sebagai Dynamic Polymorphism.
- Return type wajib sama.

DESIGN GUIDELINES

Best Practices Pemrograman Berorientasi Objek

Prinsip Liskov Substitution

Subclass harus dapat menggantikan posisi superclass tanpa merusak fungsionalitas program.

Enkapsulasi Ketat

Gunakan visibility private/protected untuk menjaga integritas data internal objek.

Komposisi over Inheritance

Gunakan pewarisan secukupnya; pertimbangkan komposisi jika hubungan bukan murni "is-a".

COMMON PITFALLS

Kesalahan Umum dalam Implementasi PBO

1. Salah Signature Method Overriding

Mengubah jumlah atau tipe parameter pada subclass, sehingga method dianggap sebagai method baru (overloading tidak disengaja) alih-alih menimpa method parent.

2. Lupa Memanggil Konstruktor Parent

Tidak menggunakan `super()` di baris pertama konstruktor anak, yang menyebabkan properti warisan gagal diinisialisasi.

ADVANCED TOPICS

Penanganan Objek Dinamis Saat Runtime

Dynamic Dispatch

Mekanisme di mana keputusan method mana yang akan dieksekusi diambil pada saat runtime (bukan compile-time) berdasarkan objek aktual yang diacu.

Keleluasaan Prototype JavaScript

JavaScript menangani hal ini secara alami melalui prototype chain dan class syntax modern ES6.



Efisiensi Eksekusi

Memungkinkan pengelolaan memori dan struktur program yang sangat modular serta adaptif.

MODULE CONCLUSION

Kesimpulan Akhir Modul Pembelajaran

Pemodelan Class Diagram

Menyediakan cetak biru logis dengan visibility (-/#/+) yang jelas.

Polimorfisme Dinamis

Melalui Overloading dan Overriding untuk penanganan objek fleksibel.

Studi Kasus JavaScript

Penerapan nyata konsep PBO secara aplikatif dalam bahasa pemrograman.



Terima Kasih

Eksplorasi mendalam Modul Inheritance, Polimorfisme, dan Studi Kasus JavaScript telah selesai. Mari kita buka sesi diskusi dan tanya jawab.

Sesi Tanya Jawab & Diskusi Terbuka