Laporan Modul 6: Inheritance

Mata Kuliah: Praktikum Pemrograman Berorientasi Objek
Nama: Riski Al Fatah
NIM: 2024573010036
Kelas: TI.2E

Abstrak

Laporan praktikum ini (Modul 6) berfokus pada konsep fundamental Inheritance (Pewarisan) dalam Pemrograman Berorientasi Objek (OOP) di Java. Tujuan praktikum ini adalah untuk memahami dan mengimplementasikan berbagai jenis dan mekanisme pewarisan untuk meningkatkan code reusability dan membangun hierarki kelas yang logis. Metodologi yang digunakan adalah melalui empat studi kasus praktikum. Praktikum pertama membahas Single Inheritance menggunakan kelas Person dan Student. Praktikum kedua mendalami Method Overriding dan penggunaan kata kunci super pada studi kasus Vehicle dan Car. Praktikum ketiga mengilustrasikan dua jenis pewarisan yang lebih kompleks: Multilevel Inheritance (melalui rantai Animal → Mammal → Dog) dan Hierarchical Inheritance (melalui cabang Mammal → Dog dan Cat). Praktikum terakhir mengintegrasikan semua konsep, termasuk abstraksi dan polimorfisme, dalam sebuah studi kasus Sistem Manajemen Perpustakaan (LibraryItem → Book, DVD, Magazine). Hasil dari keseluruhan praktikum ini menunjukkan penerapan inheritance yang efektif untuk mengurangi duplikasi kode, memungkinkan polimorfisme, dan menciptakan struktur program yang terorganisir dan mudah diperluas.

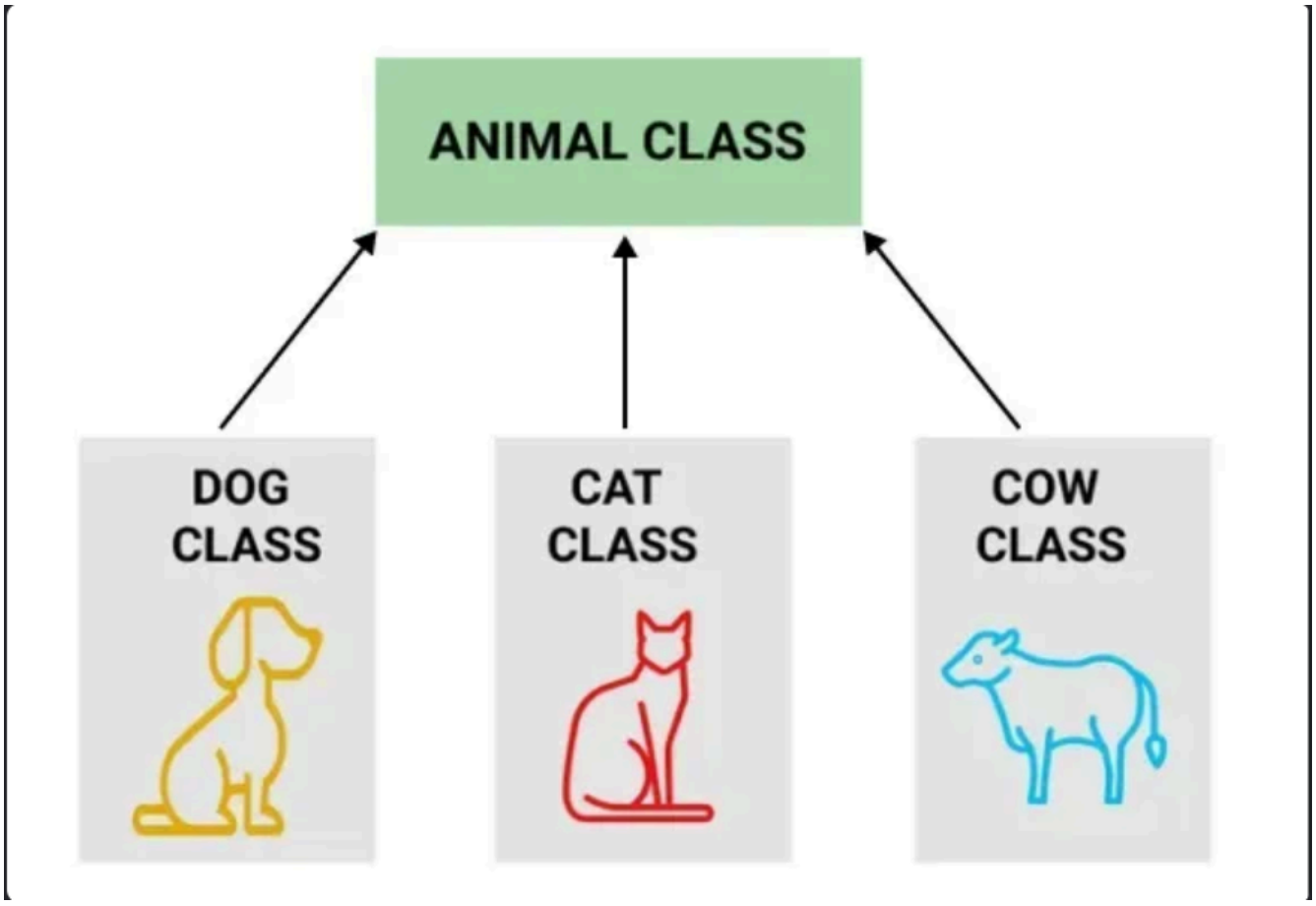
Pendahuluan

Inheritance (Pewarisan) adalah salah satu prinsip fundamental dalam Object-Oriented Programming (OOP) yang memungkinkan sebuah class (subclass/child class) mewarisi sifat dan perilaku dari class lain (superclass/parent class). Dengan inheritance, kita dapat menghindari duplikasi kode dan meningkatkan reusability.

Tujuan Inheritance

1. Code Reusability - Menggunakan kembali kode yang sudah ada tanpa menulis ulang.
2. Method Overriding - Memungkinkan subclass mengimplementasikan ulang method dari parent class.
3. Polymorphism - Memungkinkan objek subclass diperlakukan sebagai objek superclass.
4. Extensibility - Memperluas fungsionalitas class yang sudah ada.

5. Hierarchical Classification - Membuat hubungan hierarki antar class.



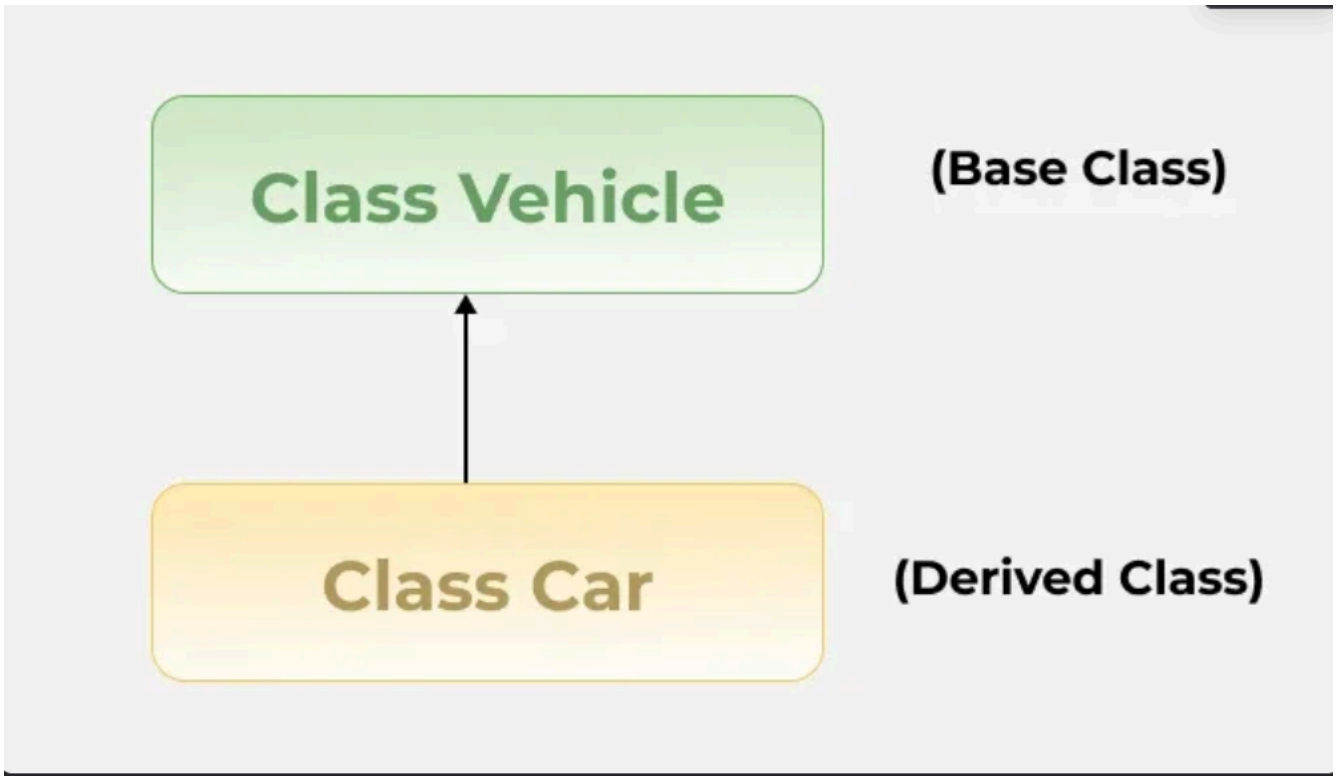
Cara Implementasi

1. Gunakan kata kunci extends untuk mewarisi dari sebuah class.
2. Subclass dapat mengakses anggota (fields dan methods) yang bersifat protected dan public dari superclass.
3. Subclass dapat mengoverride method dari superclass.
4. Gunakan kata kunci super untuk mengakses anggota superclass.

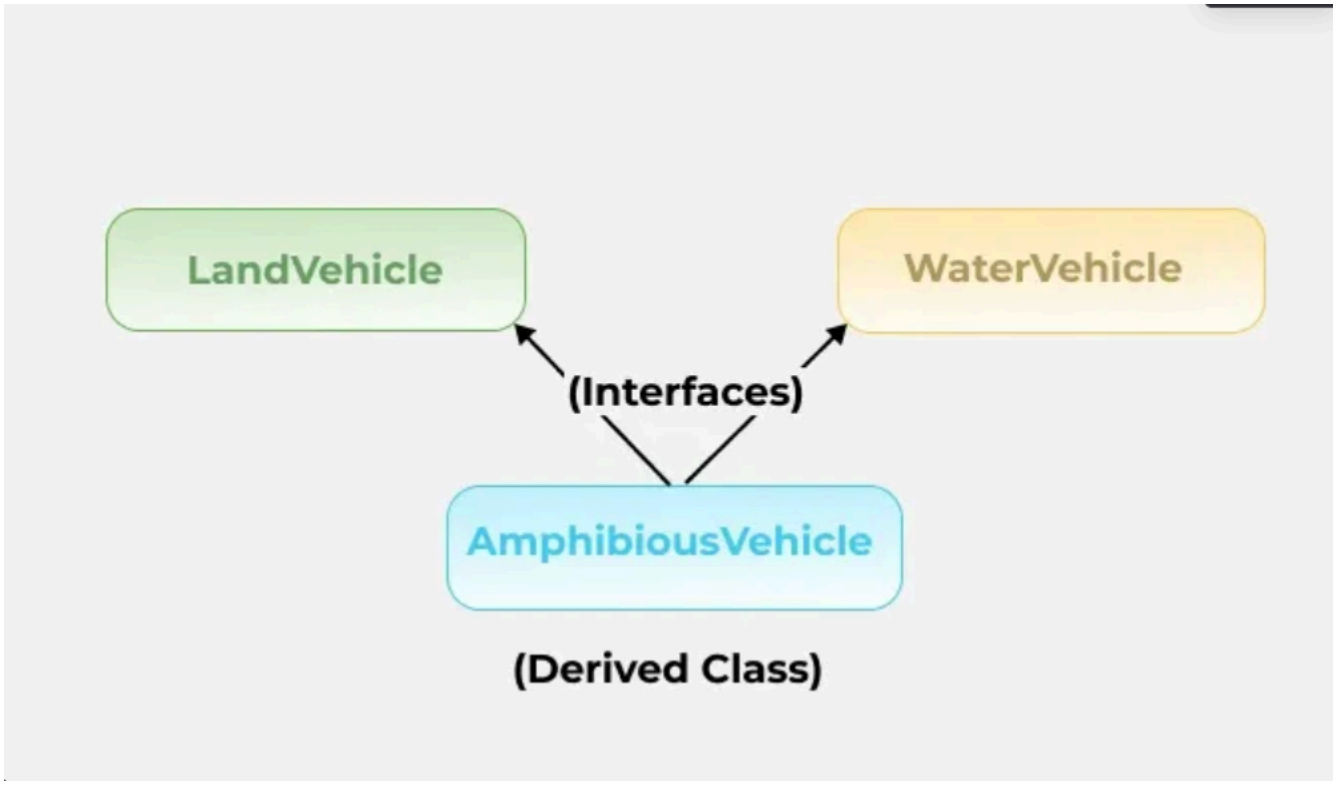
Jenis-jenis Inheritance:

1. Single Inheritance Single inheritance terjadi ketika sebuah class turunan hanya mewarisi dari satu class induk. Ini adalah bentuk pewarisan yang paling sederhana, di mana class turunan mendapatkan semua metode dan properti

dari satu class induk saja.

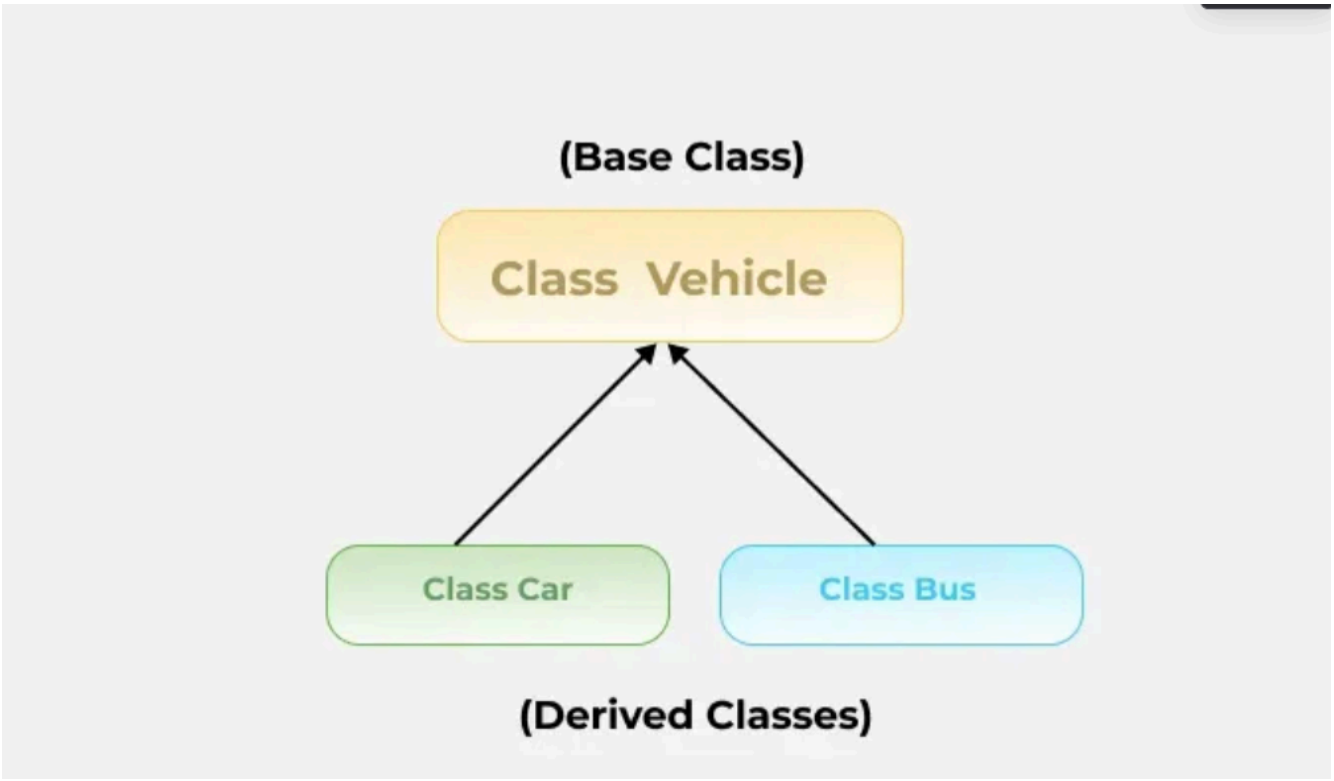


2. Multiple Inheritance Multiple inheritance memungkinkan sebuah class turunan untuk mewarisi dari lebih dari satu class induk. Artinya, class turunan dapat menggabungkan fungsionalitas dari beberapa class induk. Multiple inheritance lebih kompleks dan harus ditangani dengan hati-hati untuk menghindari masalah seperti diamond problem, di mana class turunan mewarisi dari dua class induk yang memiliki class dasar sama.

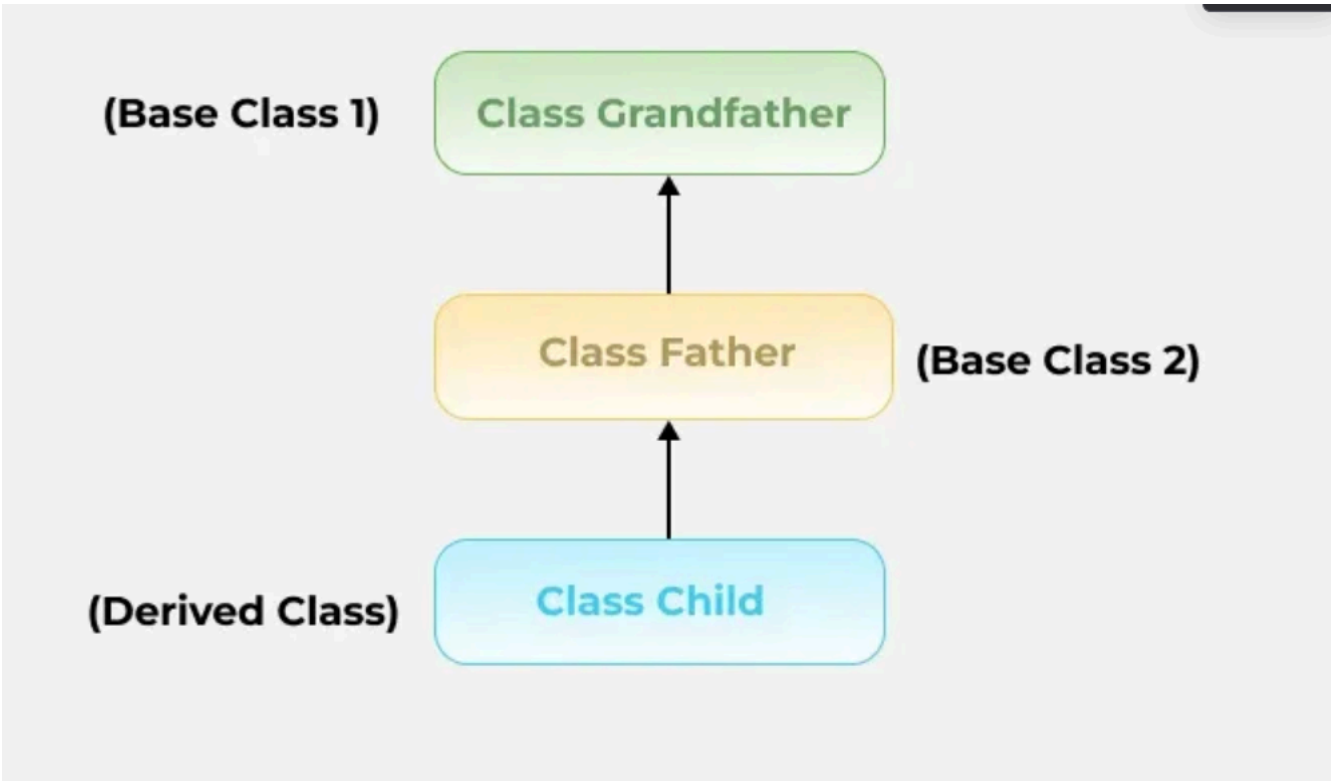


3. Hierarchical Inheritance Hierarchical inheritance terjadi saat beberapa class turunan mewarisi dari satu class induk yang sama. Jenis ini sering digunakan untuk mengelompokkan fungsionalitas yang serupa ke dalam satu class

induk, sementara memungkinkan variasi dalam class turunan.

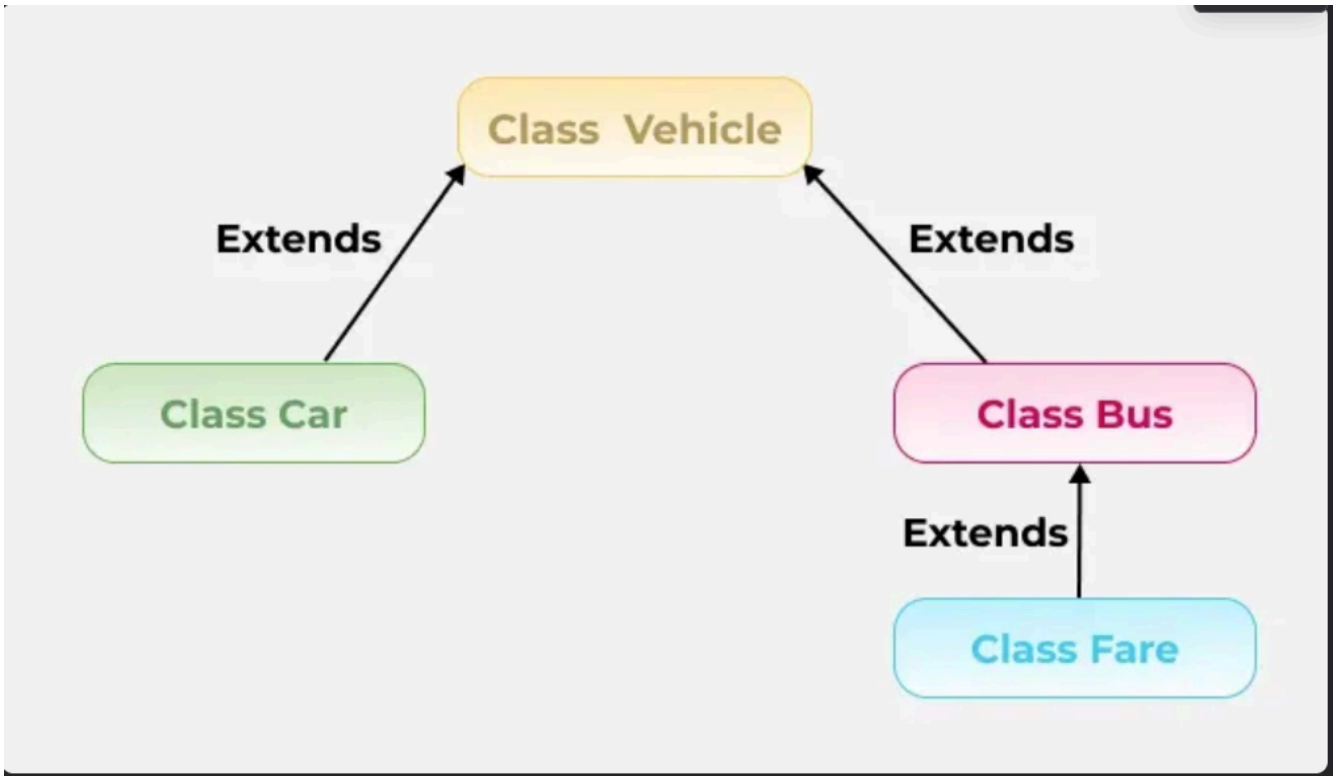


4. Multilevel Inheritance Dalam multilevel inheritance, sebuah class turunan mewarisi dari class turunan lain, sementara class yang sendiri mewarisi dari class induk. Jenis ini menciptakan "rantai" pewarisan. Misalnya, class C mewarisi dari class B, dan class B mewarisi dari class A.



5. Hybrid Inheritance Hybrid inheritance adalah kombinasi dari dua atau lebih jenis inheritance yang sebelumnya sudah disebutkan di atas. Kondisi ini sering terjadi dalam sistem yang lebih kompleks, di mana berbagai bentuk

inheritance dipakai bersama-sama untuk mencapai fleksibilitas dan efisiensi yang diinginkan.



Praktikum 1: Memahami Single Inheritance

Single Inheritance (Pewarisan Tunggal) adalah konsep fundamental dalam Pemrograman Berbasis Objek (OOP) di mana sebuah kelas turunan (disebut subclass atau child class) hanya diizinkan untuk mewarisi properti (atribut) dan perilaku (metode) dari satu kelas induk (disebut superclass atau parent class). Konsep ini menciptakan hierarki yang jelas dan linear, yang sering digambarkan sebagai hubungan "is-a" (adalah sebuah). Sebagai contoh, jika kelas Dog mewarisi dari kelas Animal, maka Dog adalah sebuah Animal. Dog akan mendapatkan semua karakteristik yang dimiliki Animal (seperti metode eat() atau sleep()), tetapi ia tidak bisa sekaligus mewarisi dari kelas lain, misalnya kelas Robot. Keuntungan utama dari model ini adalah kesederhanaan; ini membuat struktur kode lebih mudah dipahami, dikelola, dan menghindari kompleksitas serta potensi konflik (seperti "Diamond Problem") yang bisa timbul jika sebuah kelas diizinkan memiliki lebih dari satu induk (seperti dalam multiple inheritance). Bahasa pemrograman seperti Java dan C# secara ketat menerapkan model Single Inheritance ini.

Contoh Code Sigle Inheritance:

Class Person sebagai superclass:

```java
package modul_6.praktikum_1;

public class Person {  2 usages  1 inheritor  new *
    protected String name;  4 usages
    protected int age;  2 usages

    public Person(String name, int age) {  22 usages  new *
        this.name = name;
        this.age = age;
    }

    public void displayInfo() {  1 usage  new *
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }

    public void greet() {  2 usages  1 override  new *
        System.out.println("Hello, I am a person.");
    }
}
```

Class Student sebagai subclass yang mewarisi Person: teks tidak dikenali Clas InheritanceTest untuk testing:

```java
package modul_6.praktikum_1;

public class InheritanceTest {  new *
    public static void main(String[] args) {  new *
        Student student = new Student( name: "Alice",  age: 20,  studentId: "S12345");

        // Memanggil method dari superclass
        student.displayInfo();

        // Memanggil method dari subclass
        student.study();

        // Memanggil overridden method
        student.greet();

        // Polymorphism: Student sebagai Person
        Person person = new Student( name: "Bob",  age: 22,  studentId: "S67890");
        person.greet(); // Memanggil method yang di-override
    }
}
```

Output:

```
Name: Riski
Age: 20
Riski  is studying.
Hello, I am a student named Riski
Hello, I am a student named Al Fatah

Process finished with exit code 0
```

Analisa:

1. Person

- Tujuan: Ini adalah Superclass (atau kelas induk/induk). Kelas ini bertindak sebagai cetak biru (blueprint) dasar untuk mendefinisikan "Person" (Orang) secara umum.

- Atribut (Fields):

protected String name;

protected int age;

Kedua atribut ini menggunakan protected, yang berarti mereka dapat diakses langsung oleh kelas Person sendiri dan oleh kelas anak (subclass) yang mewarisinya (seperti Student).

- Constructor:

public Person(String name, int age): Ini adalah constructor yang akan dipanggil saat objek Person dibuat. Ini mengharuskan name dan age disediakan untuk menginisialisasi atribut.

- Method (Perilaku):

displayInfo(): Method ini mencetak nama dan umur.

greet(): Method ini menyediakan sapaan umum untuk seorang "Person".

Praktikum 2: Method Overriding dan Kata Kunci super

Method overriding memungkinkan subclass memberikan implementasi spesifik untuk method yang sudah didefinisikan di superclass. Kata kunci super digunakan untuk mengakses anggota superclass.

Aturan Method Overriding:

1. Method harus memiliki nama dan parameter yang sama
2. Return type harus sama atau subtype (covariant return type)
3. Access modifier tidak boleh lebih restriktif
4. Tidak bisa override method yang final atau static

Penggunaan super:

1. super() - Memanggil constructor superclass
2. super.methodName() - Memanggil method superclass
3. super.variableName - Mengakses variable superclass

Tujuan: Memahami cara melakukan method overriding dan penggunaan kata kunci super.

Contoh Code Method Overriding dan Kata Kunci super:

Class Vehicle sebagai superclass:

```java
package modul_6.praktikum_2;

public class Vehicle {  2 usages  1 inheritor  new *
    protected String brand;  2 usages
    protected int speed;  2 usages

    public Vehicle(String brand, int speed) {  3 usages  new *
        this.brand = brand;
        this.speed = speed;
    }

    public void start() {  3 usages  1 override  new *
        System.out.println("Vehicle is starting...");
    }

    public void displayInfo() {  1 override  new *
        System.out.println("Brand: " + brand);
        System.out.println("Speed: " + speed + " km/h");
    }
}
```

Class Car sebagai subclass yang mewarisi Vehicle:

```java
package modul_6.praktikum_2;

public class Car extends Vehicle {  3 usages  new *
    private int numberOfDoors;  2 usages

    public Car(String brand, int speed, int numberOfDoors) {  3 usages  new *
        super(brand, speed); // Memanggil constructor superclass
        this.numberOfDoors = numberOfDoors;
    }

    @Override  3 usages  new *
    public void start() {
        super.start(); // Memanggil method start dari superclass
        System.out.println("Car engine is running smoothly");
    }

    @Override  new *
    public void displayInfo() {
        super.displayInfo(); // Memanggil method displayInfo dari superclass
        System.out.println("Number of doors: " + numberOfDoors);
    }

    public void honk() {  1 usage  new *
        System.out.println("Beep beep!");
    }
}
```

Class OverrideTest untuk testing:

```java
package modul_6.praktikum_1;

public class Student extends Person {  3 usages  new *
    private String studentId;  1 usage

    public Student(String name, int age, String studentId) {  3 usages  new *
        super(name, age); // Memanggil constructor superclass
        this.studentId = studentId;
    }

    public void study() {  1 usage  new *
        System.out.println(name + " is studying.");
    }

    @Override  2 usages  new *
    public void greet() {
        System.out.println("Hello, I am a student named " + name);
    }
}
```

Output:

```
"C:\Program Files\Amazon Corretto\jdk21.0.8_9\bi
Vehicle is starting...
Car engine is running smoothly
Brand: Mazda
Speed: 250 km/h
Number of doors: 4
Beep beep!
Vehicle is starting...
Car engine is running smoothly
Brand: Porshe
Speed: 1000 km/h
Number of doors: 2

Process finished with exit code 0
```

Analisa: Cara Method Overriding Bekerja Method Overriding (Penimpaan Metode) adalah mekanisme di mana subclass (kelas anak) menyediakan implementasi spesifik untuk sebuah method yang sudah didefinisikan di superclass (kelas induk).

Dalam Kode Anda:

Vehicle (superclass) mendefinisikan public void start() dan public void displayInfo().

Car (subclass) meng-override kedua method tersebut. Anda menandainya dengan anotasi @Override, yang merupakan praktik terbaik untuk memberi tahu compiler bahwa Anda bermaksud menimpa method induk.

Cara Bekerja:

Di OverrideTest.java, Anda membuat Car car = new Car("Mazda", 250, 4);.

Ketika Anda memanggil car.start(), Java Virtual Machine (JVM) tidak melihat ke kelas Vehicle terlebih dahulu. Ia melihat ke tipe objek sebenarnya, yaitu Car.

JVM menemukan bahwa Car memiliki method start()-nya sendiri (yang di-override).

Oleh karena itu, ia menjalankan method start() dari kelas Car, yang menghasilkan output:

Vehicle is starting... Car engine is running smoothly Hal yang sama persis terjadi untuk car.displayInfo(), ia menjalankan versi displayInfo() dari Car.

2. Penggunaan Kata Kunci super untuk Mengakses Superclass Kata kunci super digunakan oleh subclass untuk merujuk pada bagian superclass-nya. Di kode Anda, super digunakan dalam dua cara penting di dalam Car.java:

A. Memanggil Constructor Superclass:

Kode: super(brand, speed);

Lokasi: Ada di dalam constructor Car.

Penjelasan: Sebuah Car adalah sebuah Vehicle. Saat Anda membuat Car, Anda juga harus membuat bagian Vehicle-nya. Baris super(brand, speed); memanggil constructor dari Vehicle (public Vehicle(String brand, int speed)) untuk menginisialisasi atribut brand dan speed yang diwarisi. Ini harus menjadi baris pertama di dalam constructor subclass.

B. Memanggil Method Superclass:

Kode: super.start(); dan super.displayInfo();

Lokasi: Ada di dalam method start() dan displayInfo() yang di-override di Car.

Penjelasan: Ini adalah cara Anda untuk menambahkan fungsionalitas, bukan menggantinya sepenuhnya.

Dalam start(), super.start(); terlebih dahulu menjalankan kode dari Vehicle.start() (mencetak "Vehicle is starting..."). Setelah itu, ia menjalankan kode baru dari Car (mencetak "Car engine is running smoothly").

Jika Anda menghapus baris super.start();, maka outputnya hanya "Car engine is running smoothly".

3. Konsep Polymorphism dalam Inheritance Polymorphism (secara harfiah "banyak bentuk") adalah kemampuan objek untuk diperlakukan sebagai instance dari superclass-nya. Ini memungkinkan Anda menulis kode yang lebih fleksibel.

Dalam Kode Anda:

Kode: Vehicle vehicle = new Car("Porshe", 1000, 2);

Penjelasan: Ini adalah inti dari polymorphism.

Tipe Referensi: Variabel vehicle memiliki tipe Vehicle (superclass). Ini seperti Anda memiliki "remote control" untuk Vehicle.

Tipe Objek Aktual: Objek yang dibuat new Car(...) adalah Car (subclass).

Ini diizinkan karena Car adalah sebuah Vehicle.

Cara Bekerja (Dynamic Method Dispatch):

Kode: vehicle.start(); dan vehicle.displayInfo();

Penjelasan: Meskipun "remote control" Anda adalah tipe Vehicle (yang memiliki tombol start dan displayInfo), Java cukup pintar saat runtime (saat program berjalan) untuk melihat "alat" apa yang sebenarnya terhubung ke remote itu.

Java melihat objek aktual-nya adalah Car.

Oleh karena itu, ketika Anda menekan tombol start() pada "remote" vehicle, Java akan menjalankan method start() yang telah di-override dari kelas Car, bukan dari kelas Vehicle. Inilah mengapa outputnya tetap versi Car.

Penting: Anda tidak bisa memanggil vehicle.honk();. Meskipun objeknya adalah Car (yang punya method honk()), "remote control" (tipe referensi Vehicle) tidak memiliki tombol "honk". Compiler akan error karena Vehicle tidak mendefinisikan method honk().

Praktikum 3: Multilevel dan Hierarchical Inheritance

Multilevel inheritance (bertingkat) adalah pewarisan berantai (A mewariskan ke B, lalu B mewariskan ke C). Ini seperti hubungan Kakek -> Ayah -> Anak, di mana si Anak mewarisi sifat dari Ayah dan Kakeknya.

Hierarchical inheritance (hierarkis) adalah pewarisan bercabang (A mewariskan ke B, A juga mewariskan ke C). Ini seperti satu Orang Tua (A) yang memiliki banyak anak (B dan C), di mana setiap anak mewarisi sifat yang sama dari orang tua itu.

Contoh Code Multilevel dan Hierarchical Inheritance:
Class Animal sebagai superclass:

```java
package modul_6.praktikum_3;

public class Animal {  3 usages  3 inheritors  new *
    protected String name;  8 usages

    public Animal(String name) {  6 usages  new *
        this.name = name;
    }

    public void eat() {  3 usages  2 overrides  new *
        System.out.println(name + " is eating.");
    }

    public void sleep() {  2 usages  new *
        System.out.println(name + " is sleeping.");
    }
}
```

Class Mammal yang mewarisi Animal (multilevel inheritance):

```java
package modul_6.praktikum_3;

public class Mammal extends Animal {  2 usages  2 inheritors  new *
    protected String furColor;  1 usage

    public Mammal(String name, String furColor) {  5 usages  new *
        super(name);
        this.furColor = furColor;
    }

    public void giveBirth() {  2 usages  new *
        System.out.println(name + " is giving birth to live young.");
    }
}
```

Class Dog yang mewarisi Mammal (multilevel inheritance):

```java
package modul_6.praktikum_3;

public class Dog extends Mammal {  3 usages  new *
    private String breed;  1 usage

    public Dog(String name, String furColor, String breed) {  6 usages  new *
        super(name, furColor);
        this.breed = breed;
    }

    public void bark() {  1 usage  new *
        System.out.println(name + " is barking: Woof woof!");
    }

    @Override  3 usages  new *
    public void eat() {
        System.out.println(name + " the dog is eating dog food.");
    }
}
```

Class Cat yang mewarisi Mammal (hierarchical inheritance):

```java
package modul_6.praktikum_3;

public class Cat extends Mammal {  3 usages  new *
    private boolean isIndoor;  1 usage

    public Cat(String name, String furColor, boolean isIndoor) {  3 usages  new *
        super(name, furColor);
        this.isIndoor = isIndoor;
    }

    public void meow() {  1 usage  new *
        System.out.println(name + " is meowing: Meow meow!");
    }

    @Override  3 usages  new *
    public void eat() {
        System.out.println(name + " the cat is eating cat food.");
    }
}
```

Class InheritanceTypeTest untuk testing:

```java
public class InheritanceTypeTest {  new *
    public static void main(String[] args) {  new *
        Dog dog = new Dog( name: "Buddy",  furColor: "Brown",  breed: "Golden Retriever");
        dog.eat(); // Dari Animal, di-override di Dog
        dog.sleep(); // Dari Animal
        dog.giveBirth(); // Dari Mammal
        dog.bark(); // Dari Dog

        System.out.println();

        // Hierarchical inheritance test
        Cat cat = new Cat( name: "Whiskers",  furColor: "White",  isIndoor: true);
        cat.eat(); // Dari Animal, di-override di Cat
        cat.sleep(); // Dari Animal
        cat.giveBirth(); // Dari Mammal
        cat.meow(); // Dari Cat

        System.out.println();

        // Polymorphism dengan hierarchical inheritance
        Animal[] animals = {new Dog( name: "Max",  furColor: "Black",  breed: "Labrador"),
                new Cat( name: "Luna",  furColor: "Gray",  isIndoor: false)};

        for (Animal animal : animals) {
            animal.eat(); // Akan memanggil method yang sesuai dengan objek sebenarnya
        }
    }
}
```

OutPut

```
"C:\Program Files\Amazon Corretto\jdk21.0.8_9\bin\java.exe"
Buddy the dog is eating dog food.
Buddy is sleeping.
Buddy is giving birth to live young.
Buddy is barking: Woof woof!

Whiskers the cat is eating cat food.
Whiskers is sleeping.
Whiskers is giving birth to live young.
Whiskers is meowing: Meow meow!

Max the dog is eating dog food.
Luna the cat is eating cat food.

Process finished with exit code 0
```

Analisa

1. Analisis Setiap Kode Animal.java (Kelas Dasar/Induk Paling Atas) Ini adalah kelas induk utama (superclass) atau "root" dari hierarki. Ia mendefinisikan properti (name) dan metode (eat(), sleep()) yang paling umum dan akan dimiliki oleh semua turunannya.

Mammal.java (Kelas Menengah/Induk) Kelas ini mewarisi (extends) dari Animal. Ini adalah contoh pewarisan. Ia menambahkan fungsionalitas khusus mamalia (furColor, giveBirth()). Penting: Mammal adalah anak dari Animal, tetapi ia juga sekaligus induk bagi Dog dan Cat.

Dog.java (Kelas Turunan/Anak) Kelas ini mewarisi (extends) dari Mammal. Ia menambahkan perilaku spesifik anjing (breed, bark()). Kelas ini juga melakukan override terhadap metode eat() yang diwarisi dari Animal, memberikan implementasi khusus untuk anjing.

Cat.java (Kelas Turunan/Anak) Kelas ini juga mewarisi (extends) dari Mammal. Sama seperti Dog, ia menambahkan perilaku spesifik kucing (isIndoor, meow()) dan juga melakukan override pada metode eat().

InheritanceTypeTest.java (Kelas Penguji) Kelas ini tidak mewarisi dari apapun. Tujuannya hanya satu: menggunakan kelas-kelas di atas untuk membuktikan bahwa konsep pewarisan dan polimorfisme bekerja. Ini adalah tempat di mana objek (dog, cat, animals[]) dibuat dan metode mereka dipanggil.

2. Cara Multilevel Inheritance Membentuk Rantai Multilevel inheritance (pewarisan bertingkat/berantai) terjadi ketika sebuah kelas mewarisi dari kelas lain, yang juga mewarisi dari kelas lain lagi.

Dalam kode Anda, rantai ini terbentuk dengan jelas:

Animal → Mammal → Dog

Mammal mewarisi dari Animal.

Dog mewarisi dari Mammal.

Ini terbukti di InheritanceTypeTest.java saat Anda membuat objek Dog:

Dog dog = new Dog("Buddy", "Brown", "Golden Retriever"); Objek dog ini sekarang memiliki akses ke metode dari tiga level sekaligus:

dog.bark(); (milik Dog sendiri)

dog.giveBirth(); (diwarisi dari Mammal - induknya)

dog.sleep(); (diwarisi dari Animal - "kakek"-nya)

Hal yang sama berlaku untuk rantai Animal → Mammal → Cat.

3. Cara Hierarchical Inheritance Memungkinkan Pewarisan Bercabang Hierarchical inheritance (pewarisan hierarkis/bercabang) terjadi ketika satu kelas induk diwarisi oleh lebih dari satu kelas anak.

Dalam kode Anda, ini berpusat pada kelas Mammal:

Mammal bertindak sebagai induk tunggal.

Dog adalah kelas anak pertama (public class Dog extends Mammal).

Cat adalah kelas anak kedua (public class Cat extends Mammal).

Dog dan Cat adalah "saudara" (siblings) dalam hierarki ini. Keduanya sama-sama mewarisi sifat dan perilaku dari Mammal (seperti giveBirth()) dan juga dari Animal (seperti sleep()), tetapi mereka tidak saling berhubungan satu sama lain. Dog tidak bisa meow(), dan Cat tidak bisa bark().

4. Cara Polymorphism Bekerja dengan Hierarki Polymorphism (polimorfisme) berarti "banyak bentuk". Dalam OOP, ini memungkinkan kita memperlakukan objek dari kelas turunan seolah-olah mereka adalah objek dari kelas induknya.

Ini ditunjukkan dengan sempurna di bagian akhir InheritanceTypeTest.java:

a. Upcasting (Menyimpan Anak di Tipe Induk)

Animal[] animals = {new Dog("Max", "Black", "Labrador"), new Cat("Luna", "Gray", false)}; Di sini, Anda membuat array bertipe Animal. Namun, Anda mengisinya dengan objek Dog dan Cat. Ini legal karena (berkat inheritance) Dog adalah Animal dan Cat adalah Animal.

b. Dynamic Method Dispatch (Perilaku Berbeda saat Runtime)

for (Animal animal : animals) { animal.eat(); }

Ini adalah inti dari polimorfisme. Anda memanggil metode eat() pada variabel animal (yang tipenya Animal). Namun, Java cukup pintar untuk tahu:

Saat animal merujuk ke objek Dog, ia akan memanggil metode eat() yang ada di kelas Dog (yang di-@Override).

Saat animal merujuk ke objek Cat, ia akan memanggil metode eat() yang ada di kelas Cat (yang juga di-@Override).

Inilah mengapa outputnya adalah:

Max the dog is eating dog food. Luna the cat is eating cat food. Bukan is eating. dari kelas Animal. Metode yang dipanggil bergantung pada objek aslinya saat runtime, bukan pada tipe variabelnya saat compile time.

Praktikum 4: Sistem Manajemen Perpustakaan Sederhana

Menerapkan konsep inheritance dalam project real-world sederhana untuk mengelola sistem perpustakaan.

Deskripsi Project:

Kita akan membuat sistem manajemen perpustakaan sederhana yang memiliki berbagai jenis item (buku, majalah, DVD) dengan karakteristik yang berbeda namun memiliki beberapa kesamaan. **Contoh Code** Class LibraryItem sebagai superclass:

```
package modul_6.praktikum_4;

public abstract class LibraryItem {  11 usages  3 inheritors  new *
    protected String itemId;  5 usages
    protected String title;  11 usages
    protected int year;  5 usages
    protected boolean isAvailable;  9 usages

    public LibraryItem(String itemId, String title, int year) {  7 usages  new *
        this.itemId = itemId;
        this.title = title;
        this.year = year;
        this.isAvailable = true;
    }

    // Getter methods
    public String getItemId() { return itemId; }  1 usage  new *
    public String getTitle() { return title; }  no usages  new *
    public int getYear() { return year; }  no usages  new *
    public boolean isAvailable() { return isAvailable; }  3 usages  new *

    // Setter methods
    public void setAvailable(boolean available) { isAvailable = available; }  no usages  new *

    // Abstract method yang harus diimplementasikan subclass
    public abstract void displayInfo();  3 implementations  new *

    // Concrete method yang bisa digunakan semua subclass
    public void borrowItem() {  1 usage  new *
```

```
        if (isAvailable) {
            isAvailable = false;
            System.out.println(title + " berhasil dipinjam");
        } else {
            System.out.println(title + " sedang tidak tersedia");
        }
    }

    public void returnItem() {  1 usage  new *
        isAvailable = true;
        System.out.println(title + " berhasil dikembalikan");
    }
}
```

Class Book yang mewarisi LibraryItem:

```
package modul_6.praktikum_4;                                                    ⚠4 ✓9

public class Book extends LibraryItem {  3 usages  new *
    private String author;  2 usages
    private String isbn;  2 usages
    private int numberOfPages;  2 usages

    public Book(String itemId, String title, int year, String author, String isbn, int numberOfPages) {  4 usages  new
        super(itemId, title, year);
        this.author = author;
        this.isbn = isbn;
        this.numberOfPages = numberOfPages;
    }

    @Override  new *
    public void displayInfo() {
        System.out.println("BUKU");
        System.out.println("--------------------------");
        System.out.println("ID: " + itemId);
        System.out.println("Judul: " + title);
        System.out.println("Penulis: " + author);
        System.out.println("Tahun: " + year);
        System.out.println("ISBN: " + isbn);
        System.out.println("Jumlah Halaman: " + numberOfPages);
        System.out.println("Status: " + (isAvailable ? "Tersedia" : "Dipinjam"));
        System.out.println("--------------------------");
    }
```

```java
    // Method khusus Book
    public void readSample() {  no usages  new *
        System.out.println("Membaca sample dari buku: " + title);
    }
}
```

Class Magazine yang mewarisi LibraryItem:

```java
public class Magazine extends LibraryItem {  2 usages  new *
    private String publisher;  2 usages
    private int issueNumber;  2 usages
    private String category;  2 usages

    public Magazine(String itemId, String title, int year, String publisher, int issueNumber, String category) {
        super(itemId, title, year);
        this.publisher = publisher;
        this.issueNumber = issueNumber;
        this.category = category;
    }

    @Override  new *
    public void displayInfo() {
        System.out.println("--------- MAJALAH ---------");
        System.out.println("ID: " + itemId);
        System.out.println("Judul: " + title);
        System.out.println("Penerbit: " + publisher);
        System.out.println("Tahun: " + year);
        System.out.println("Edisi: " + issueNumber);
        System.out.println("Kategori: " + category);
        System.out.println("Status: " + (isAvailable ? "Tersedia" : "Dipinjam"));
        System.out.println("---------------------------");
    }

    // Method khusus Magazine
    public void browseArticles() {  no usages  new *
        System.out.println("Menelusuri artikel dalam majalah: " + title);
    }
```

Class DVD yang mewarisi LibraryItem:

```java
public class DVD extends LibraryItem {  2 usages  new *
    private String director;  2 usages
    private int duration; // dalam menit  2 usages
    private String genre;  2 usages

    public DVD(String itemId, String title, int year, String director, int duration, String genre) {  4 usages  new *
        super(itemId, title, year);
        this.director = director;
        this.duration = duration;
        this.genre = genre;
    }

    @Override  new *
    public void displayInfo() {
        System.out.println("---- DVD ----");
        System.out.println("ID: " + itemId);
        System.out.println("Judul: " + title);
        System.out.println("Sutradara: " + director);
        System.out.println("Tahun: " + year);
        System.out.println("Durasi: " + duration + " menit");
        System.out.println("Genre: " + genre);
        System.out.println("Status: " + (isAvailable ? "Tersedia" : "Dipinjam"));
        System.out.println("---------------");
    }

    // Method khusus DVD
    public void playTrailer() {  no usages  new *
        System.out.println("Memutar trailer DVD: " + title);
    }
```

Class LibraryManagementSystem sebagai main class:

```java
package modul_6.praktikum_4;

import java.util.ArrayList;
import java.util.Scanner;

public class LibraryManagementSystem {  new *
    static ArrayList<LibraryItem> libraryItems = new ArrayList<>();  12 usages
    static Scanner scanner = new Scanner(System.in);  22 usages

    public static void main(String[] args) {  new *
        initializeData();
        int choice;
        do {
            displayMenu();
            System.out.print("Pilih menu (1-7): ");
            choice = scanner.nextInt();
            scanner.nextLine(); // consume newline

            switch (choice) {
                case 1:
                    displayAllItems();
                    break;
                case 2:
                    addItem();
                    break;
                case 3:
                    borrowItem();
                    break;
```

```
                case 4:
                    returnItem();
                    break;
                case 5:
                    displayAvailableItems();
                    break;
                case 6:
                    displayBorrowedItems();
                    break;
                case 7:
                    System.out.println("Terima kasih telah menggunakan sistem ini.");
                    break;
                default:
                    System.out.println("Pilihan tidak valid!");
            }
            System.out.println();
        } while (choice != 7);
    }

    private static void displayMenu() {  1 usage  new *
        System.out.println("--- SISTEM MANAJEMEN PERPUSTAKAAN ---");
        System.out.println("1. Tampilkan Semua Item");
        System.out.println("2. Tambah Item Baru");
        System.out.println("3. Pinjam Item");
        System.out.println("4. Kembalikan Item");
        System.out.println("5. Tampilkan Item Tersedia");
        System.out.println("6. Tampilkan Item Dipinjam");
        System.out.println("7. Keluar");
    }

    private static void initializeData() {  1 usage  new *
        libraryItems.add(new Book( itemId: "B001",  title: "The Lord of the Rings",  year: 1954,  author: "J.R.R. Tolkien",
                isbn: "978-0618640157",  numberOfPages: 1178));
        libraryItems.add(new Book( itemId: "B002",  title: "Harry Potter and the Sorcerer's Stone",  year: 1997,  author: "J.K. Rowling",
                isbn: "978-0590353427",  numberOfPages: 309));
        libraryItems.add(new DVD( itemId: "D001",  title: "Inception",  year: 2010,  director: "Christopher Nolan",
                duration: 148,  genre: "Sci-Fi"));
        libraryItems.add(new Magazine( itemId: "M001",  title: "National Geographic",  year: 2023,
                publisher: "National Geographic Society",  issueNumber: 202305,  category: "Sains"));
    }

    private static void displayAllItems() {  1 usage  new *
        System.out.println("\n--- DAFTAR SEMUA ITEM ---");
        if (libraryItems.isEmpty()) {
            System.out.println("Belum ada item di perpustakaan.");
        } else {
            for (LibraryItem item : libraryItems) {
                item.displayInfo();
            }
        }
    }
```

```java
private static void displayAvailableItems() {  1 usage  new *
    System.out.println("\n--- DAFTAR ITEM TERSEDIA ---");
    boolean found = false;
    for (LibraryItem item : libraryItems) {
        if (item.isAvailable()) {
            item.displayInfo();
            found = true;
        }
    }
    if (!found) {
        System.out.println("Semua item sedang dipinjam.");
    }
}

private static void displayBorrowedItems() {  1 usage  new *
    System.out.println("\n--- DAFTAR ITEM DIPINJAM ---");
    boolean found = false;
    for (LibraryItem item : libraryItems) {
        if (!item.isAvailable()) {
            item.displayInfo();
            found = true;
        }
    }
    if (!found) {
        System.out.println("Tidak ada item yang sedang dipinjam.");
    }
}
```

```java
private static LibraryItem findItemById(String itemId) {  2 usages  new *
    for (LibraryItem item : libraryItems) {
        if (item.getItemId().equalsIgnoreCase(itemId)) {
            return item;
        }
    }
    return null;
}

private static void borrowItem() {  1 usage  new *
    System.out.println("\n--- PINJAM ITEM ---");
    System.out.print("Masukkan ID item yang akan dipinjam: ");
    String itemId = scanner.nextLine();
    LibraryItem item = findItemById(itemId);

    if (item != null) {
        item.borrowItem();
    } else {
        System.out.println("Item dengan ID " + itemId + " tidak ditemukan.");
    }
}

private static void returnItem() {  1 usage  new *
    System.out.println("\n--- KEMBALIKAN ITEM ---");
    System.out.print("Masukkan ID item yang akan dikembalikan: ");
    String itemId = scanner.nextLine();
    LibraryItem item = findItemById(itemId);
```

```
        if (item != null) {
            item.returnItem();
        } else {
            System.out.println("Item dengan ID " + itemId + " tidak ditemukan.");
        }
    }

    private static void addItem() {  1 usage  new *
        System.out.println("\n--- TAMBAH ITEM BARU ---");
        System.out.println("Pilih tipe item (1: Buku, 2: DVD, 3: Majalah): ");
        int type = scanner.nextInt();
        scanner.nextLine(); // consume newline

        System.out.print("Masukkan ID (misal B003): ");
        String itemId = scanner.nextLine();
        System.out.print("Masukkan Judul: ");
        String title = scanner.nextLine();
        System.out.print("Masukkan Tahun: ");
        int year = scanner.nextInt();
        scanner.nextLine();

        switch (type) {
            case 1: // Buku
                System.out.print("Masukkan Penulis: ");
                String author = scanner.nextLine();
                System.out.print("Masukkan ISBN: ");
                String isbn = scanner.nextLine();
```

```
                System.out.print("Masukkan Jumlah Halaman: ");
                int pages = scanner.nextInt();
                scanner.nextLine();
                libraryItems.add(new Book(itemId, title, year, author, isbn, pages));
                break;
            case 2: // DVD
                System.out.print("Masukkan Sutradara: ");
                String director = scanner.nextLine();
                System.out.print("Masukkan Durasi (menit): ");
                int duration = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Masukkan Genre: ");
                String genre = scanner.nextLine();
                libraryItems.add(new DVD(itemId, title, year, director, duration, genre));
                break;
            case 3: // Majalah
                System.out.print("Masukkan Penerbit: ");
                String publisher = scanner.nextLine();
                System.out.print("Masukkan Nomor Edisi: ");
                int issue = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Masukkan Kategori: ");
                String category = scanner.nextLine();
                libraryItems.add(new Magazine(itemId, title, year, publisher, issue, category));
                break;
            default:
                System.out.println("Tipe item tidak valid.");
                return;
        }
        System.out.println("Item baru berhasil ditambahkan.");
    }
}
```

OutPut

```
--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 1

--- DAFTAR SEMUA ITEM ---
BUKU
--------------------------
ID: B001
Judul: The Lord of the Rings
Penulis: J.R.R. Tolkien
Tahun: 1954
ISBN: 978-0618640157
Jumlah Halaman: 1178
Status: Tersedia
--------------------------
BUKU
--------------------------
ID: B002
```

```
Judul: Harry Potter and the Sorcerer's Stone
Penulis: J.K. Rowling
Tahun: 1997
ISBN: 978-0590353427
Jumlah Halaman: 309
Status: Tersedia
---------------------------
---- DVD ----
ID: D001
Judul: Inception
Sutradara: Christopher Nolan
Tahun: 2010
Durasi: 148 menit
Genre: Sci-Fi
Status: Tersedia
--------------
--------- MAJALAH ---------
ID: M001
Judul: National Geographic
Penerbit: National Geographic Society
Tahun: 2023
Edisi: 202305
Kategori: Sains
Status: Tersedia
```

```
--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 2

--- TAMBAH ITEM BARU ---
Pilih tipe item (1: Buku, 2: DVD, 3: Majalah):
2
Masukkan ID (misal B003): B006
Masukkan Judul: fatah
Masukkan Tahun: 2006
Masukkan Sutradara: riski
Masukkan Durasi (menit): 30
Masukkan Genre: romance
Item baru berhasil ditambahkan.

--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
```

```
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 3

--- PINJAM ITEM ---
Masukkan ID item yang akan dipinjam: B006
fatah berhasil dipinjam

--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 4

--- KEMBALIKAN ITEM ---
Masukkan ID item yang akan dikembalikan: B006
fatah berhasil dikembalikan

--- SISTEM MANAJEMEN PERPUSTAKAAN ---
```

```
--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 5

--- DAFTAR ITEM TERSEDIA ---
BUKU
--------------------------
ID: B001
Judul: The Lord of the Rings
Penulis: J.R.R. Tolkien
Tahun: 1954
ISBN: 978-0618640157
Jumlah Halaman: 1178
Status: Tersedia
--------------------------
BUKU
--------------------------
ID: B002
Judul: Harry Potter and the Sorcerer's Stone
```

```
Penulis: J.K. Rowling
Tahun: 1997
ISBN: 978-0590353427
Jumlah Halaman: 309
Status: Tersedia
---------------------------
---- DVD ----
ID: D001
Judul: Inception
Sutradara: Christopher Nolan
Tahun: 2010
Durasi: 148 menit
Genre: Sci-Fi
Status: Tersedia
---------------
--------- MAJALAH ---------
ID: M001
Judul: National Geographic
Penerbit: National Geographic Society
Tahun: 2023
Edisi: 202305
Kategori: Sains
Status: Tersedia
---------------------------
---- DVD ----
```

```
---- DVD ----
ID: B006
Judul: fatah
Sutradara: riski
Tahun: 2006
Durasi: 30 menit
Genre: romance
Status: Tersedia
--------------

--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 6

--- DAFTAR ITEM DIPINJAM ---
Tidak ada item yang sedang dipinjam.

--- SISTEM MANAJEMEN PERPUSTAKAAN ---
1. Tampilkan Semua Item
```

```
2. Tambah Item Baru
3. Pinjam Item
4. Kembalikan Item
5. Tampilkan Item Tersedia
6. Tampilkan Item Dipinjam
7. Keluar
Pilih menu (1-7): 7
Terima kasih telah menggunakan sistem ini.


Process finished with exit code 0
```

Analisa

1. LibraryItem.java Peran: Kelas Induk Abstrak (Blueprint / Cetak Biru).

Analisis Kunci:

Abstraction (Abstraksi): Dideklarasikan sebagai public abstract class. Ini berarti Anda tidak bisa membuat objek new LibraryItem(). Kelas ini hanya ada sebagai template/konsep umum tentang "apa itu item perpustakaan".

Properti Umum: Ia mendefinisikan properti (itemId, title, year, isAvailable) yang pasti dimiliki oleh semua jenis item, baik itu buku, majalah, atau DVD.

Metode Konkret (Code Reuse): Ia menyediakan metode borrowItem() dan returnItem(). Logika untuk meminjam dan mengembalikan adalah sama untuk semua item, jadi kode ini ditulis sekali di kelas induk dan dapat digunakan oleh semua kelas anak.

Metode Abstrak (Kontrak): public abstract void displayInfo(); adalah "kontrak" atau paksaan. LibraryItem tidak tahu cara menampilkan info secara spesifik (karena buku punya penulis, DVD punya sutradara). Dengan menjadikannya abstract, ia memaksa setiap kelas anak (seperti Book, DVD) untuk wajib menulis implementasi @Override dari metode displayInfo() versi mereka sendiri.

2. Book.java Peran: Kelas Turunan Konkret.

Analisis Kunci:

Inheritance (Pewarisan): public class Book extends LibraryItem berarti Book adalah sebuah LibraryItem. Ia secara otomatis mewarisi semua properti (seperti title) dan metode (seperti borrowItem()) dari LibraryItem.

Spesialisasi: Book menambahkan properti unik yang hanya dimiliki buku (author, isbn, numberOfPages) dan metode unik (readSample()).

Implementasi Kontrak: Book memenuhi "kontrak" dari LibraryItem dengan menyediakan implementasi @Override untuk metode displayInfo(). Versi ini mencetak informasi spesifik buku, termasuk penulis dan ISBN.

Konstruktor: super(itemId, title, year); adalah perintah untuk memanggil konstruktor kelas induk (LibraryItem) guna menginisialisasi properti umum.

3. Magazine.java Peran: Kelas Turunan Konkret.

Analisis Kunci:

Inheritance (Pewarisan): Sama seperti Book, public class Magazine extends LibraryItem berarti Magazine adalah sebuah LibraryItem dan mewarisi semua sifat dan perilaku umumnya.

Spesialisasi: Ia menambahkan properti unik majalah (publisher, issueNumber, category) dan metode unik (browseArticles()).

Implementasi Kontrak: Ia juga memenuhi kontrak dengan menyediakan implementasi @Override untuk displayInfo() yang menampilkan detail spesifik majalah.

Hierarki: Ini menunjukkan Hierarchical Inheritance. Baik Book maupun Magazine sama-sama mewarisi dari satu induk yang sama, yaitu LibraryItem.

4. DVD.java Peran: Kelas Turunan Konkret.

Analisis Kunci:

Inheritance (Pewarisan): public class DVD extends LibraryItem berarti DVD adalah sebuah LibraryItem.

Spesialisasi: Ia menambahkan properti unik DVD (director, duration, genre) dan metode unik (playTrailer()).

Implementasi Kontrak: Ia juga menyediakan @Override untuk displayInfo() yang diformat khusus untuk menampilkan detail DVD.

Hierarki: DVD juga merupakan bagian dari Hierarchical Inheritance bersama Book dan Magazine, semuanya bercabang dari LibraryItem.

5. LibraryManagementSystem.java Peran: Kelas Utama / Driver (Penggerak Aplikasi).

Analisis Kunci:

Titik Masuk: Berisi public static void main(String[] args), yang merupakan titik awal eksekusi seluruh program.

Polymorphism (Polimorfisme) : Ini adalah konsep kunci yang ditunjukkan di sini, terutama pada baris:

static ArrayList libraryItems = new ArrayList<>(); Perhatikan bahwa ArrayList ini bertipe LibraryItem (kelas induk abstrak), tetapi di dalam metode initializeData(), kita mengisinya dengan objek anak (new Book(...), new DVD(...), new Magazine(...)). Ini bisa dilakukan karena berkat inheritance, Book adalah LibraryItem, DVD adalah LibraryItem, dst.

Polymorphism in Action: Keajaiban polimorfisme terlihat jelas di dalam metode displayAllItems():

for (LibraryItem item : libraryItems) { item.displayInfo(); // <-- INI AJAIBNYA } Meskipun kita memanggil displayInfo() pada variabel item yang bertipe LibraryItem, Java (saat runtime) cukup pintar untuk tahu: "Oh, item ini sebenarnya objek Book? Oke, panggil displayInfo() milik Book." atau "Oh, item ini sebenarnya objek DVD? Panggil displayInfo() milik DVD."

Manajemen: Kelas ini bertugas mengelola seluruh logika aplikasi, seperti menampilkan menu, menangani input user (Scanner), dan memanggil metode-metode yang sesuai (seperti item.borrowItem() dari LibraryItem atau item.displayInfo() dari kelas anaknya).

Kesimpulan

Berdasarkan empat praktikum yang telah dilaksanakan, dapat ditarik beberapa kesimpulan mengenai konsep inheritance (pewarisan) dalam Pemrograman Berorientasi Objek:

1. Inheritance adalah mekanisme fundamental OOP yang memungkinkan sebuah kelas (subclass) mewarisi properti dan metode dari kelas lain (superclass), seperti yang ditunjukkan pada Praktikum 1 (Student mewarisi Person). Mekanisme ini secara efektif mengurangi duplikasi kode dan meningkatkan code reusability.
2. Method Overriding, seperti pada Praktikum 2 (Car meng-override metode Vehicle), memberikan kemampuan bagi subclass untuk menyediakan implementasi spesifik dari metode yang diwarisi. Penggunaan kata kunci super menjadi krusial untuk dapat mengakses konstruktor dan metode orisinal dari superclass saat melakukan overriding.
3. Terdapat berbagai pola pewarisan yang dapat diterapkan sesuai kebutuhan. Praktikum 3 mengilustrasikan dua pola utama: Multilevel Inheritance (pewarisan berantai seperti Animal → Mammal → Dog) dan Hierarchical Inheritance (pewarisan bercabang seperti Mammal → Dog dan Mammal → Cat).
4. Inheritance adalah fondasi utama untuk Polymorphism (polimorfisme). Praktikum 4 (Sistem Perpustakaan) membuktikan hal ini dengan jelas. Dengan membuat kelas induk abstrak (LibraryItem), kita dapat membuat berbagai kelas turunan spesifik (Book, DVD, Magazine). Melalui polimorfisme, objek-objek yang berbeda tipe ini dapat dikelola secara seragam dalam satu koleksi (misal, ArrayList) namun akan memanggil metode (displayInfo()) yang sesuai dengan implementasi override di kelas aslinya.

Secara keseluruhan, Modul 6 ini menunjukkan bahwa inheritance, method overriding, dan polymorphism bukanlah konsep yang terpisah, melainkan serangkaian mekanisme yang bekerja bersama-sama untuk menciptakan kode yang terstruktur, fleksibel, mudah dikelola, dan mudah dikembangkan (extensible).

Referensi

Oracle Corporation. (2024). The Java™ Tutorials: Inheritance. Diakses pada 6 November 2025, dari https://docs.oracle.com/javase/tutorial/java/IandI/inheritance.html
Petani Kode. (2019). Belajar Java: Memahami Konsep Pewarisan (Inheritance). Diakses pada 6 November 2025, dari https://www.petanikode.com/java-inheritance/