LAPORAN PRAKTIKUM XI

FUNDAMENTAL OF PROGRAMMING WORKSHOP



DOSEN PENGAJAR

Indrawati, SST.MT

**DISUSUN OLEH**

**Kelompok I:**

DIKI CANDRA

**Kelas : TRKJ-1B**

POLITEKNIK NEGERI LHOKSEUMAWE

JURUSAN TEKNOLOGI KOMUNIKASI DAN KOMPUTER

PRODI D-IV TEKNOLOGI REKAYASA KOMPUTER DAN JARINGAN

2022/2023

# Daftar Isi

[Daftar Isi 1](#_Toc733816627)

[I. Tujuan Pratikum 2](#_Toc1089787921)

[II. Dasar Teori 3](#_Toc667130064)

[III. Alat dan Bahan 5](#_Toc1266756961)

[IV. Langkah Percobaan 5](#_Toc257835326)

[V. Hasil dan Analisa 6](#_Toc1995054301)

[Kesimpulan 8](#_Toc1445406063)

# Tujuan Pratikum

* 1. Mengidentifikasi macam-macam fungsi dari Java
  2. Mengimplementasikan bahasa pemprograman java
  3. Membiasakan diri menggunakan Java

# Dasar Teori

A. Pengertian Pewarisan

**Inheritance** atau *Pewarisan/Penurunan* adalah konsep pemrograman dimana sebuah *class* dapat ‘*menurunkan*’ ***property***dan ***method*** yang dimilikinya kepada ***class***lain. Konsep ***inheritance***digunakan untuk memanfaatkan fitur ‘*code reuse*’ untuk menghindari duplikasi kode program.

Konsep *inheritance* membuat sebuah struktur atau ‘*hierarchy*’ ***class***dalam kode program**.** Class yang akan ‘*diturunkan*’ bisa disebut sebagai ***class induk (parent class), super class,***atau***base class***. Sedangkan class yang ‘*menerima penurunan*’ bisa disebut sebagai ***class anak (child class), sub class, derived class***atau***heir class***.

Tidak semua *property* dan *method* dari *class* induk akan diturunkan. *Property* dan *method* dengan hak akses ***private***, tidak akan diturunkan kepada class anak. Hanya *property* dan method dengan hak akses ***protected***dan ***public***saja yang bisa diakses dari *class* anak.

Suatu class yang mempunyai class turunan dinamakan parent class atau base class. Sedangkan class turunan itu sendiri seringkali disebut subclass atau child class. Suatu subclass dapat mewarisi apa-apa yang dipunyai oleh parent class.

Karena suatu subclass dapat mewarisi apa apa yang dipunyai oleh parent class-nya, maka member dari suatu subclass adalah terdiri dari apa-apa yang ia punyai dan juga apa-apa yang ia warisi dari class parent-nya.

Kesimpulannya, boleh dikatakan bahwa suatu subclass adalah tidak lain hanya memperluas (extend) parent class-nya.

Dari hirarki diatas dapat dilihat bahwa, semakin kebawah, class akan semakin bersifat spesifik. Class mamalia memiliki seluruh sifat yang dimiliki oleh binatang, demikian halnya juga macan , kucing, Paus dan Monyet memiliki seluruh sifat yang diturunkan dari class mamalia.

Dengan konsep ini, karakteristik yang dimiliki oleh class binatang cukup didefinisikan didefinisikan dalam class binatang saja.

Class mamalia tidak perlu mendefinisikan ulang apa yang telah dimiliki oleh class binatang, karena sebagai class turunannya, ia akan mendapatkan karakteristik dari class binatang secara otomatis. Demikian juga dengan class macan, kucing, Paus dan monyet, hanya perlu mendefinisikan karakteristik yang spesifik dimiliki oleh class-nya masing-masing.

Dengan memanfaatkan konsep pewarisan ini dalam pemrograman, maka hanya perlu mendefinisikan karakteristik yang lebih umum akan didapatkan dari class darimana ia diturunkan.

**ISTILAH INHERITANCE**

**Extends** Keyword ini harus kita tambahkan pada definisi class yang menjadi subclass.

**Superclass** Superclass digunakan untuk menunjukkan hirarki class yang berarti class dasar dari subclass/class anak.

**Subclass** Subclass adalah class anak atau turunan secara hirarki dari superclass.

**Super** Keyword ini digunakan untuk memanggil konstruktor dari superclass atau menjadi variabel yang mengacu pada superclass.

**Methode Overriding** Pendefinisian ulang method yang sama pada subclass

# Alat dan Bahan

* 1. Laptop
  2. NeatBeans

# Langkah Percobaan

1. **Program 1**

Pewarisan Antar Kelas

1. // Superclass
2. class Kendaraan {
3. String merek;
4. void bergerak() {
5. System.out.println("Kendaraan bergerak.");
6. }
7. }
8. // Subclass
9. class Mobil extends Kendaraan {
10. int jumlahPintu;
11. void klakson() {
12. System.out.println("Mobil berbunyi klakson.");
13. }
14. }
15. public class Main {
16. public static void main(String[] args) {
17. Mobil mobil = new Mobil();
18. mobil.merek = "Toyota";
19. mobil.jumlahPintu = 4;
20. mobil.bergerak(); // Memanggil metode dari superclass
21. mobil.klakson(); // Memanggil metode dari subclass
22. }
23. }

**Program 2**

Pewarisan Antar Interface

// Superinterface

interface Hewan {

void makan();

}

// Subinterface

interface Herbivora extends Hewan {

void makanTumbuhan();

}

// Implementasi

class Kuda implements Herbivora {

public void makan() {

System.out.println("Kuda sedang makan rumput.");

}

public void makanTumbuhan() {

System.out.println("Kuda merupakan herbivora.");

}

}

public class Main {

public static void main(String[] args) {

Kuda kuda = new Kuda();

kuda.makan(); // Memanggil metode dari interface Hewan

kuda.makanTumbuhan(); // Memanggil metode dari interface Herbivora

}

}

1. **Program 3**

Pewarisan Antar Kelas Abstrak

// Kelas abstrak

abstract class Bentuk {

String warna;

abstract double hitungLuas();

void setWarna(String warna) {

this.warna = warna;

}

}

// Subclass

class Persegi extends Bentuk {

double sisi;

double hitungLuas() {

return sisi \* sisi;

}

}

public class Main {

public static void main(String[] args) {

Persegi persegi = new Persegi();

persegi.setWarna("Merah");

persegi.sisi = 5;

System.out.println("Warna Persegi: " + persegi.warna);

System.out.println("Luas Persegi: " + persegi.hitungLuas());

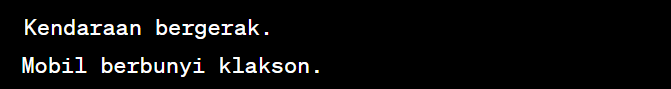
}

}

# Hasil dan Analisa

Hasil:

1. Pewarisan Antar Kelas



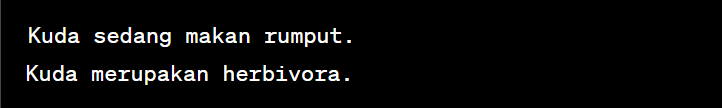
Analisa:

Program tersebut membuat objek **mobil** menggunakan kelas **Mobil**. Setelah itu, atribut **merek** diisi dengan nilai "Toyota" dan **jumlahPintu** diisi dengan nilai 4.

Kemudian, program memanggil metode **bergerak()** dari superclass **Kendaraan** yang kemudian mencetak "Kendaraan bergerak." di konsol. Selanjutnya, program memanggil metode **klakson()** dari subclass **Mobil** yang mencetak "Mobil berbunyi klakson." di konsol.

Dengan demikian, output program menunjukkan bahwa objek **mobil** mewarisi atribut dan metode dari kelas **Kendaraan** (superclass) dan juga memiliki metode khusus yang ada di kelas **Mobil** (subclass).

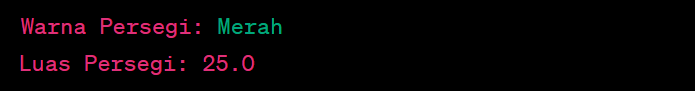
1. Memakai local variable



Program tersebut membuat objek **kuda** menggunakan kelas **Kuda**. Setelah itu, program memanggil metode **makan()** dari interface **Hewan** yang diimplementasikan oleh kelas **Kuda**, yang mencetak "Kuda sedang makan rumput." di konsol. Selanjutnya, program memanggil metode **makanTumbuhan()** dari interface **Herbivora** yang juga diimplementasikan oleh kelas **Kuda**, yang mencetak "Kuda merupakan herbivora." di konsol.

Dengan demikian, output program menunjukkan bahwa objek **kuda** dapat menggunakan metode yang didefinisikan dalam interface **Hewan** dan **Herbivora**. Melalui pewarisan antar interface, kelas **Kuda** mewarisi dan mengimplementasikan metode-metode dari kedua interface tersebut.

1. Memakai member variabel



Analisa :

Program di atas memiliki kelas abstrak **Bentuk**, yang memiliki metode abstrak **hitungLuas()** dan metode **setWarna()**. Kelas **Persegi** merupakan subclass dari **Bentuk** dan mengimplementasikan metode **hitungLuas()** sesuai dengan karakteristik persegi.

Dalam program utama, objek **persegi** dibuat menggunakan kelas **Persegi**. Kemudian, metode **setWarna()** dipanggil untuk mengatur warna persegi menjadi "Merah". Selanjutnya, atribut **sisi** diisi dengan nilai 5.

Program mencetak warna persegi menggunakan atribut **warna** dari kelas **Bentuk**. Selanjutnya, program memanggil metode **hitungLuas()** dari kelas **Persegi** untuk menghitung luas persegi dan mencetak hasilnya.

# Kesimpulan

metode pewarisan adalah fitur penting dalam pemrograman berorientasi objek yang memungkinkan kelas untuk mewarisi atribut dan metode dari kelas lain. Dengan menggunakan pewarisan, kita dapat membuat hierarki kelas yang memungkinkan penggunaan kembali kode, mengorganisir kode secara hierarkis, dan memodelkan hubungan "is-a" antara kelas-kelas.

Beberapa kesimpulan tentang metode pewarisan dalam Java adalah sebagai berikut:

1. Pewarisan memungkinkan pembentukan hierarki kelas, di mana kelas yang lebih khusus (subclass) dapat mewarisi atribut dan metode dari kelas yang lebih umum (superclass).
2. Superclass adalah kelas yang memberikan warisan, sedangkan subclass adalah kelas yang menerima warisan.
3. Dalam Java, pewarisan antar kelas dicapai dengan menggunakan kata kunci "extends". Subclass menggambarkan spesialisasi atau varian dari superclass.
4. Dengan menggunakan pewarisan, subclass dapat mengakses atribut dan metode yang diwarisi dari superclass, serta menambahkan metode dan atribut khusus yang spesifik untuk subclass tersebut.
5. Pewarisan memungkinkan penggunaan kembali kode, karena kita dapat menggunakan implementasi yang ada dalam superclass tanpa perlu menulis ulang di setiap subclass.
6. Metode dan konstruktor dalam superclass dapat di-override di subclass, yang berarti subclass dapat memberikan implementasi khusus untuk metode tersebut. Ini memungkinkan polimorfisme, di mana objek dari subclass dapat digunakan dalam konteks superclass.
7. Pewarisan juga memungkinkan penggunaan pola "is-a" di mana subclass dianggap sebagai jenis superclass. Misalnya, subclass "Mobil" adalah jenis dari superclass "Kendaraan".
8. Dalam pewarisan, kita dapat memiliki hierarki kelas dengan tingkat multiple, di mana sebuah kelas dapat mewarisi dari beberapa superclass. Namun, di Java, hanya pewarisan tunggal yang didukung secara langsung antara kelas-kelas.

Pewarisan adalah salah satu konsep utama dalam pemrograman berorientasi objek yang memungkinkan pemodelan yang fleksibel dan penggunaan kembali kode. Dengan pemahaman yang baik tentang metode pewarisan dalam Java, kita dapat membangun hierarki kelas yang efisien dan mempermudah pengembangan aplikasi yang kompleks.