**LAPORAN RESMI**

**MODUL VI**

**INTERFACE**

**PEMROGRAMAN BERORIENTASI OBJEK**



**NAMA : ABAS PERMADANI**

**N.R.P : 230441100107**

**DOSEN : ACHMAD ZAIN NUR S.Kom M.T**

**ASISTEN : MUHAMMAD ROSYID MAULANA**

**TGL PRAKTIKUM : 22 MEI 2024**

**Disetujui : 03 Juni 2024**

**Asisten**

**MUHAMMAD ROSYID MAULANA 20.04.411.00019**

**LABORATORIUM BISNIS INTELIJEN SISTEM**

**PRODI SISTEM INFORMASI**

**JURUSAN TEKNIK INFORMATIKA**

**FAKULTAS TEKNIK**

**UNIVERSITAS TRUNOJOYO MADURA**

# **BAB I PENDAHULUAN**

## **Latar Belakang**

Java adalah bahasa pemrograman tingkat tinggi yang didesain untuk menjadi platform independen, artinya kode yang ditulis dalam Java dapat dijalankan di berbagai platform tanpa perlu modifikasi ulang. Salah satu fitur utama yang membuat Java sangat populer adalah kemampuannya dalam membangun aplikasi yang handal dan aman. Interface dalam pemrograman Java adalah sebuah konsep yang memungkinkan deklarasi metode tanpa implementasi konkret. Sebuah interface hanya mendefinisikan perilaku tanpa memberikan implementasi apa pun. Dalam Java, sebuah kelas dapat mengimplementasikan satu atau lebih interface, yang berarti kelas tersebut harus menyediakan implementasi untuk semua metode yang dideklarasikan dalam interface tersebut. Salah satu kegunaan utama dari interface dalam Java adalah untuk menciptakan hubungan kontrak antara berbagai kelas. Dengan menggunakan interface, pengembang dapat membuat kode yang lebih fleksibel dan dapat di-maintain dengan baik. Interface memungkinkan kelas-kelas yang berbeda untuk berkomunikasi satu sama lain tanpa perlu mengetahui detail implementasi masing-masing.

## **Tujuan**

1. Mahasiswa mampu memahami konsep Interface dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya.
2. mahasiswa dapat mengimplementasikan berbagai fitur dan perilaku dalam kode Java mereka secara lebih efektif.
3. Memahami konsep interface akan mempersiapkan mahasiswa untuk mengembangkan aplikasi yang lebih kompleks dan scalable dalam lingkungan nyata.
4. Memahami konsep interface memungkinkan mahasiswa untuk mengembangkan kemampuan dalam merancang solusi untuk masalah yang kompleks dalam pengembangan perangkat lunak.

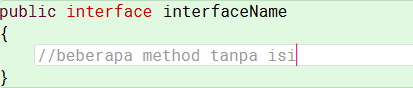
# **BAB II DASAR TEORI**

## **Konsep Interface**

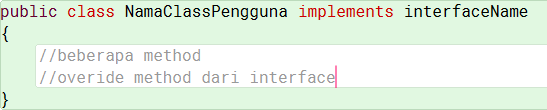
Interface merupakan suatu mekanisme dalam Java yang memungkinkan untuk berbagi konstanta atau menentukan bentuk method yang dapat digunakan oleh sejumlah class. Sebuah class dapat mengimplementasikan lebih dari satu interface.

**1. Deklarasi dan penggunaan interface:**

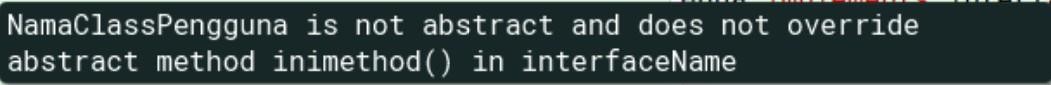
Deklarasi interface



Penggunaan interface



Ketika class Anda mencoba mengimplementasikan sebuah interface, selalu pastikan bahwa Anda mengimplementasikan semua method dari interface, jika tidak, Anda akan menemukan kesalahan ini,

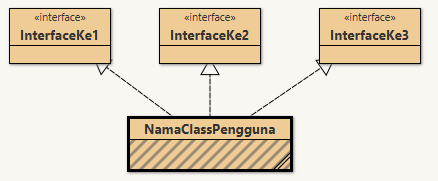


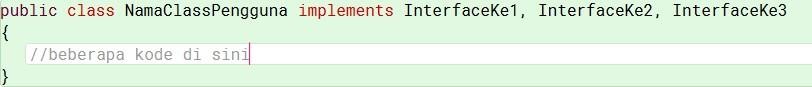
## **2.2 Hubungan dari Interface ke Class**

Class dapat mengimplementasikan sebuah interface selama kode implementasi untuk semua method yang didefinisikan dalam interface tersedia.

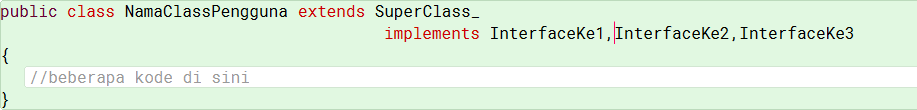
Hal lain yang perlu dicatat tentang hubungan antara interface ke class-class yaitu, class hanya dapat mengEXTEND SATU superclass, tetapi dapat mengIMPLEMENTASIkan BANYAK interface. Sebuah contoh dari sebuah class yang mengimplementasikan interface adalah,

Gambar diagramnya:

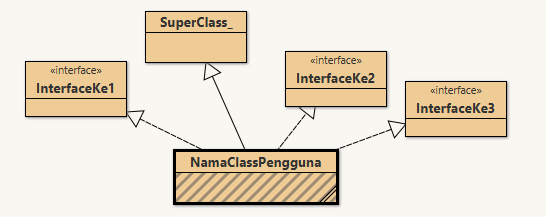




Contoh lain dari class yang meng-extend satu superclass dan mengimplementasikan sebuah interface adalah,

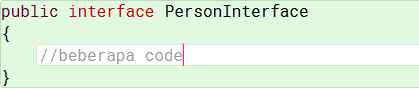


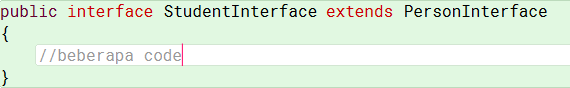
Gambar diagramnya:



## **2.3 Pewarisan Antar Interface**

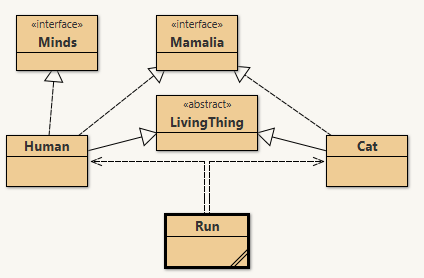
Interface bukan bagian dari hirarki class. Bagaimanapun, interface dapat mempunyai hubungan pewarisan antara mereka sendiri. Contohnya, misal kita punya dua interface StudentInterface dan PersonInterface. Jika StudentInterface meng-extend PersonInterface, maka ia akan mewariskan semua deklarasi method dalam PersonInterface.





## **2.4 Contoh Program**

Kita akan membuat superclass bernama LivingThing. class ini mempunyai method tertentu seperti breath, eat, sleep, dan walk Akan tetapi, ada beberapa method di dalam superclass yang sifatnya tidak dapat digeneralisasi. Kita ambil contoh, method walk. Tidak semua kehidupan berjalan(walk) dalam cara yang sama. Ambil manusia sebagai misal, kita manusia berjalan dengan dua kaki, dimana kehidupan lainnya seperti kucing berjalan dengan empat kaki. .kemudian kita membuat dua interface yaitu mamalia dan minds, interface mamalia berisi method characteristic dari mamalia dan interface minds berisi method mind yang akan membedakan antar manusia dan kucing.



Pada program ini ada,

2 subclass yaitu

* Class Human
* Class Cat 1

1 abstract class

* Abstract class LivingThing

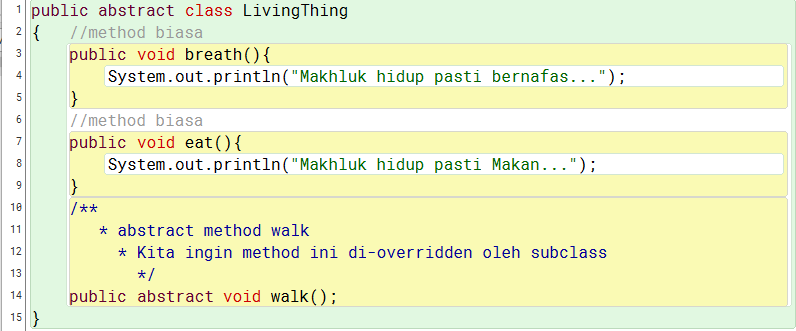
2 interface

* Interface Mamalia
* Interface Minds

1 class untuk menjalankan yaitu

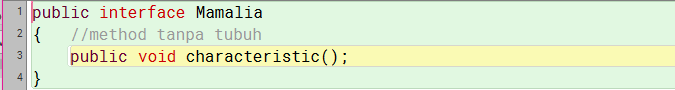
* Class Run

Membuat abstract class LivingThing:



Method walk merupakan method abstract , Ketika class meng-extend class abstract LivingThing, dibutuhkan untuk override method abstract walk(),

Membuat Interface Mamalia:



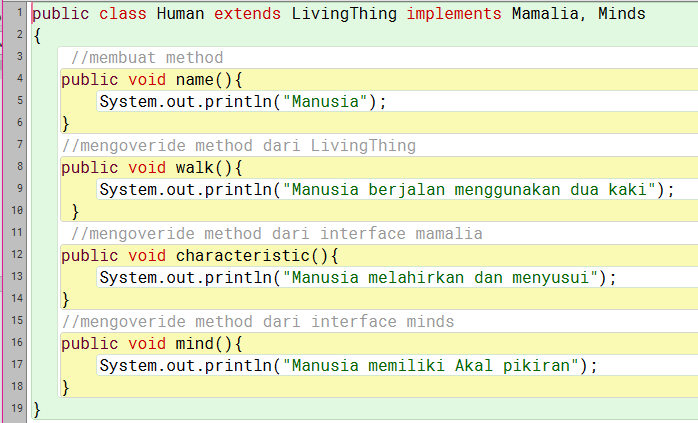
Ketika ada class yang mengimplement interface Mamalia dibutuhkan untuk override method characteristic();

Membuat interface Minds :



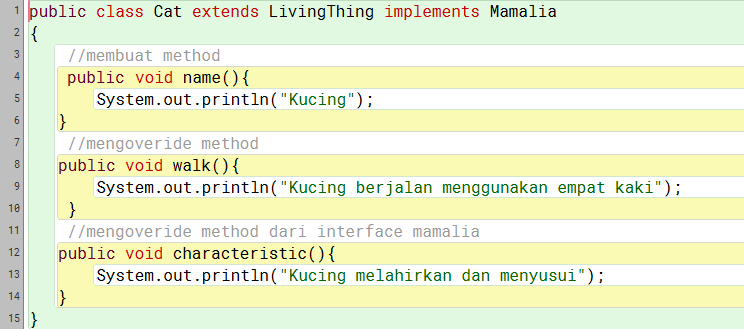
Ketika ada class yang mengimplement interface Minds dibutuhkan untuk override method mind();

Membuat Class Human :



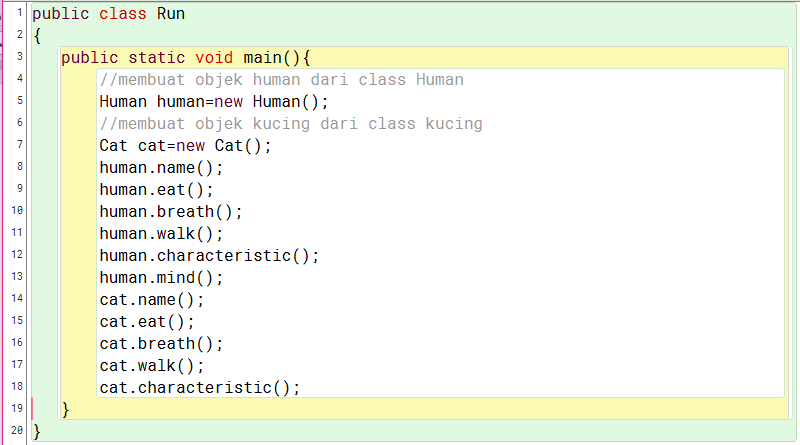
Class Human mengextend abstract class LivingThing dan harus mengoveride method walk() kemudian class Human juga mengimplements interface Mamalia dan Minds yang mengharuskan mengoveride method characteristic dari Mamalia dan method mind dari interface Minds.

Membuat Class Cat :



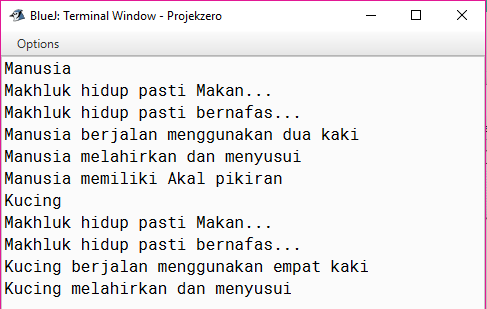
Sama seperti class Human ,class Cat ini mengextend dari abstract class LivingThing diharuskan mengoveride method walk() dan class Cat ini juga mengimplement interface Mamalia yang mengharuskan mengoveride method characteristic dari interface Mamalia

Membuat class Run yang digunakan untuk menjalankan program :



Membuat Objek dari class Human dan dari class Cat kemudian memanggil semua method dari masing – masing class.

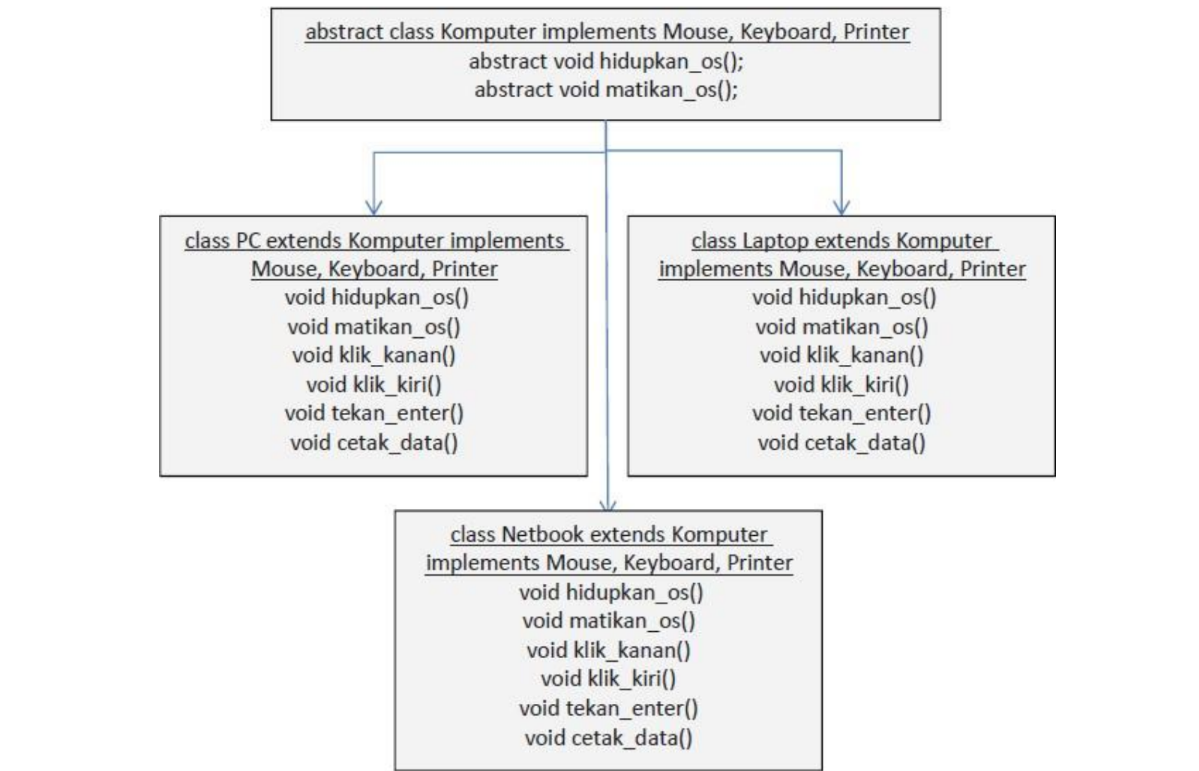
Running Program :

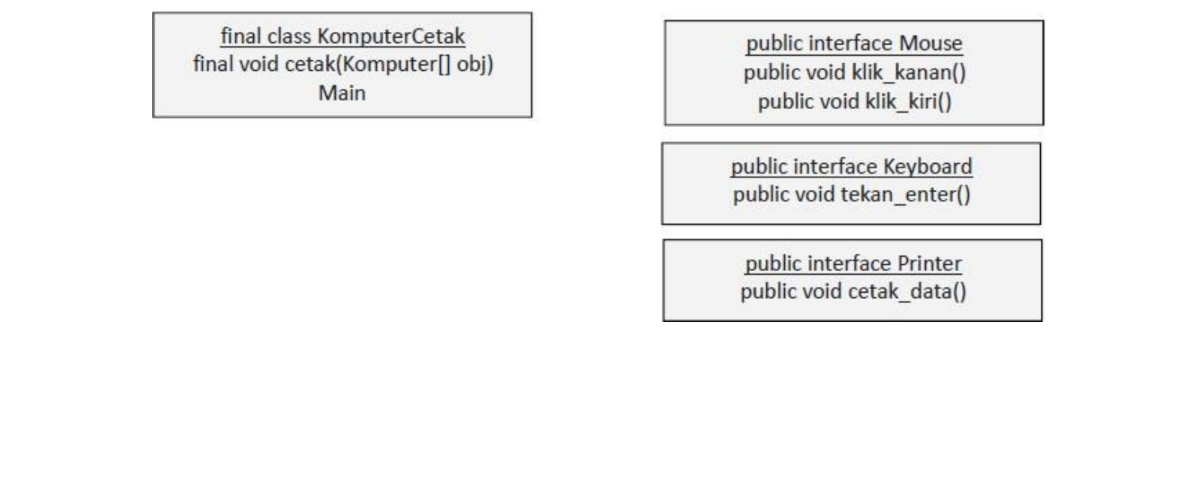


# **BAB IV IMPLEMENTASI**

## **Tugas Praktikum**

Implementasikan diagram di bawah ini menggunakan konsep interface, abstract class, dan polimorfisme





## **4.2 Source Code**

package tugas;

public abstract class Komputer {

public abstract void hidupkan\_os();

public abstract void matikan\_os();

}

package tugas;

public interface Mouse {

public void klik\_kanan();

public void klik\_kiri();

}

package tugas;

public interface Keyboard {

public void tekan\_enter();

}

package tugas;

public interface Printer {

public void cetak\_data();

}

package tugas;

public class PC extends Komputer implements Mouse, Keyboard, Printer {

@Override

public void hidupkan\_os(){

System.out.println("silahkan hidupkan PC anda dengan tombol power");

}

@Override

public void matikan\_os(){

System.out.println("silahkan matikan PC anda dengan tombol power");

}

@Override

public void klik\_kanan(){

System.out.println("klik kanan untuk sesuatu pada PC anda");

}

@Override

public void klik\_kiri(){

System.out.println("klik kiri untuk sesuatu pada PC anda");

}

@Override

public void tekan\_enter(){

System.out.println("tekan enter pada PC anda untuk memberikan space pada deskripsi");

}

@Override

public void cetak\_data(){

System.out.println("silahkan cetak data anda dengan icon print yang ada di PC anda");

System.out.println();

}

}

package tugas;

public class Laptop extends Komputer implements Mouse, Keyboard, Printer {

@Override

public void hidupkan\_os(){

System.out.println("silahkan hidupkan Laptop anda dengan tombol power");

}

@Override

public void matikan\_os(){

System.out.println("silahkan matikan Laptop anda dengan tombol power");

}

@Override

public void klik\_kanan(){

System.out.println("klik kanan untuk sesuatu pada Laptop anda");

}

@Override

public void klik\_kiri(){

System.out.println("klik kiri untuk sesuatu pada Laptop anda");

}

@Override

public void tekan\_enter(){

System.out.println("tekan enter pada Laptop anda untuk memberikan space pada deskripsi");

}

@Override

public void cetak\_data(){

System.out.println("silahkan cetak data anda dengan icon print yang ada di laptop anda");

System.out.println();

}

}

package tugas;

public class Netbook extends Komputer implements Mouse, Keyboard, Printer {

@Override

public void hidupkan\_os(){

System.out.println("silahkan hidupkan Netbook anda dengan tombol power");

}

@Override

public void matikan\_os(){

System.out.println("silahkan matikan Netbook anda dengan tombol power");

}

@Override

public void klik\_kanan(){

System.out.println("klik kanan untuk sesuatu pada Netbook anda");

}

@Override

public void klik\_kiri(){

System.out.println("klik kiri untuk sesuatu pada Netbook anda");

}

@Override

public void tekan\_enter(){

System.out.println("tekan enter pada Netbook anda untuk memberikan space pada deskripsi");

}

@Override

public void cetak\_data(){

System.out.println("silahkan cetak data anda dengan icon print yang ada di Netbook anda");

}

}

package tugas;

public class Main {

public static void main(String[] args) {

PC alat1 = new PC ();

Laptop alat2 = new Laptop ();

Netbook alat3 = new Netbook ();

alat1.hidupkan\_os();

alat1.matikan\_os();

alat1.klik\_kanan();

alat1.klik\_kiri();

alat1.tekan\_enter();

alat1.cetak\_data();

alat2.hidupkan\_os();

alat2.matikan\_os();

alat2.klik\_kanan();

alat2.klik\_kiri();

alat2.tekan\_enter();

alat2.cetak\_data();

alat3.hidupkan\_os();

alat3.matikan\_os();

alat3.klik\_kanan();

alat3.klik\_kiri();

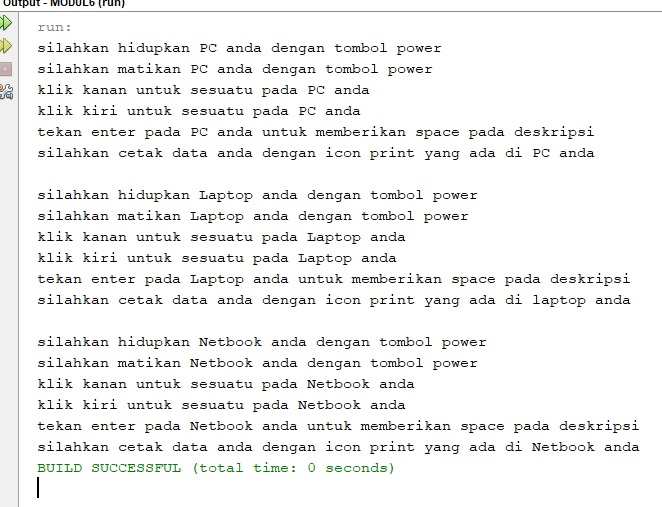
alat3.tekan\_enter();

alat3.cetak\_data();

}

}

## **4.3 Hasil Output**



## **4.4 Penjelasan**

Kode tersebut merupakan contoh implementasi konsep abstraksi dan antarmuka dalam pemrograman berorientasi objek (OOP) menggunakan bahasa pemrograman Java. Dalam kode tersebut, terdapat sebuah kelas abstrak bernama Komputer yang memiliki dua metode abstrak, yaitu hidupkan\_os() dan matikan\_os(). Selain itu, ada tiga antarmuka yaitu Mouse, Keyboard, dan Printer, yang masing-masing mendefinisikan beberapa metode tanpa implementasi. Tiga kelas, yaitu PC, Laptop, dan Netbook, mewarisi kelas Komputer dan mengimplementasikan antarmuka-antarmuka tersebut. Setiap kelas harus mengimplementasikan metode-metode yang didefinisikan oleh kelas abstrak dan antarmuka yang mereka warisi. Dalam kelas Main, dilakukan pembuatan objek dari ketiga kelas tersebut dan pemanggilan metode-metodenya. Dengan menggunakan konsep abstraksi dan antarmuka, kode tersebut memungkinkan untuk menggambarkan perilaku umum dari berbagai jenis komputer tanpa harus mengimplementasikan detailnya di semua kelas secara terpisah, memudahkan pengembangan, pemeliharaan, dan perluasan kode di masa depan.

# **BAB V PENUTUP**

## **Analisa**

Dari hasil praktikum, praktikan menganalisa bahwa memahami tentang interface dalam pemrograman Java merupakan langkah penting dalam menguasai konsep dasar Pemrograman Berorientasi Objek (OOP). Mereka menyadari bahwa interface memberikan kerangka kerja yang kuat untuk membangun kode yang modular, terstruktur, dan mudah di-maintain. Dengan memahami bagaimana interface bekerja, praktikan dapat melihat manfaatnya dalam pengembangan perangkat lunak, terutama dalam memfasilitasi komunikasi antara berbagai kelas dalam sebuah sistem. Mereka menyadari bahwa interface memungkinkan kelas-kelas yang berbeda untuk berinteraksi satu sama lain tanpa harus mengetahui detail implementasi masing-masing. Hal ini mempromosikan konsep enkapsulasi, di mana implementasi internal suatu kelas dapat disembunyikan dari kelas-kelas lain yang bergantung padanya. Selain itu, praktikan juga menyadari bahwa penggunaan interface memfasilitasi penggunaan pola desain yang efektif, seperti pola Abstract Factory dan Strategy, yang membantu dalam memecahkan masalah kompleks dalam pengembangan perangkat lunak. Dengan memahami konsep interface secara mendalam, praktikan merasa lebih percaya diri dalam mengembangkan aplikasi Java yang lebih kompleks dan scalable.

## **Kesimpulan**

1. Mahasiswa menyadari bahwa pemahaman tentang interface membantu dalam menciptakan struktur kode yang lebih teratur dan terstruktur.
2. Pemahaman tentang interface meningkatkan kemampuan analisis mereka terhadap struktur dan hubungan antar kelas dalam sebuah sistem.
3. Mahasiswa yang memahami konsep interface akan lebih siap untuk mengembangkan aplikasi yang lebih kompleks dan scalable dalam lingkungan nyata.
4. Pemahaman mahasiswa tentang interface memberikan landasan yang kuat untuk pembelajaran yang lebih lanjut dalam pengembangan perangkat lunak.