LAPORAN RESMI
MODUL VI
INTERFACE
PEMROGRAMAN BERBASIS OBJEK

NAMA	: ANISYAFAAH
N.R.P	: 220441100105
DOSEN	: FIRMANSYAH ADIPUTRA, ST., M.Cs.
ASISTEN	: KUKUH COKRO WIBOWO
TGL PRAKTIKUM	: 12 MEI 2023

Disetujui : 17 Mei 2023
Asisten

KUKUH COKRO WIBOWO
21.04.411.00102

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pemrograman Berorientasi Objek (Object Oriented Programming) adalah paradigma pemrograman yang berfokus pada konsep objek sebagai unit utama dalam pembangunan perangkat lunak. Pemrograman berorientasi objek memodelkan dunia nyata dengan mewakili objek yang memiliki atribut (data) dan perilaku (method) yang terkait dengannya. Objek juga dapat beroperasi dan berinteraksi satu sama lain melalui pertukaran pesan, yang memungkinkan komunikasi dan kerjasama diantara objek - objek tersebut.

Interface (antarmuka) dalam bahasa pemrograman Java adalah kontrak yang digunakan untuk mendefinisikan perilaku atau kemampuan yang harus dimiliki oleh kelas yang mengimplementasikannya. Interface menyediakan definisi umum untuk perilaku yang diharapkan oleh suatu objek. Interface mendefinisikan serangkaian metode yang harus diimplementasikan oleh kelas yang menggunakan interface tersebut. Dalam istilah lain, interface berfungsi sebagai blueprint untuk perilaku objek. Berikut ini adalah contoh pembuatan program menggunakan Java dengan konsep interface.

1.2 Tujuan

- Mahasiswa mampu memahami konsep Interface dalam Pemrograman Berorientasi Objek serta mampu mengimplementasikannya

BAB II
DASAR TEORI

2.1 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.

a) Deklarasi dan Penggunaan Interface :

- Deklarasi interface

```
public interface interfaceName
{
    // beberapa method tanpa isi
}
```

- Penggunaan interface

```
public class NamaClassPengguna implements interfaceName
{
    // beberapa method
    // overide method dari interface
}
```

Ketika class Anda mencoba mengimplementasikan sebuah interface, selalu pastikan bahwa Anda mengimplementasikan semua method dari interface, jika tidak, Anda akan menemukan kesalahan ini, NamaClassPengguna is not abstract and does not override abstract method inimethod() in interfaceName.

b) Hubungan dari Interface ke Class

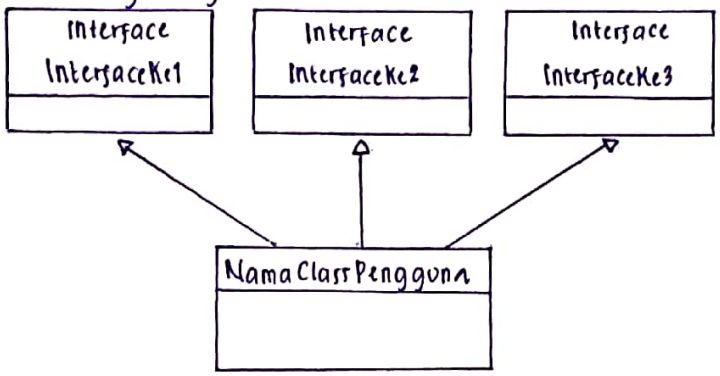
Class dapat mengimplementasikan sebuah interface selama kode implementasi untuk semua method yang didefinisikan dalam interface tersedia.

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,

```
public class NamaClassPengguna implements InterfaceKe1, InterfaceKe2,
InterfaceKe3
{
  // beberapa kode di sini
}
```

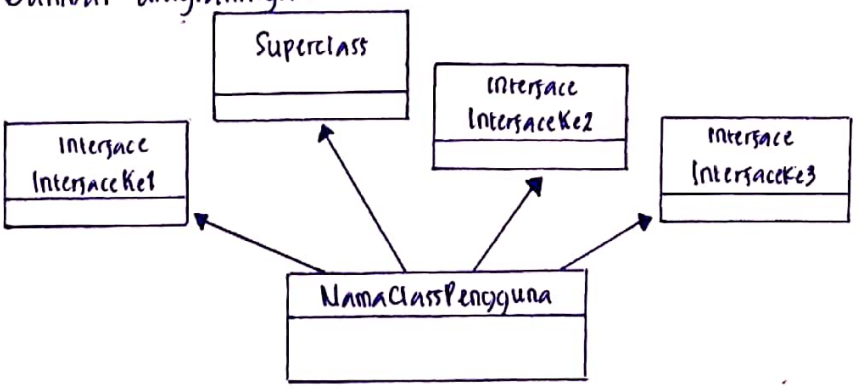
Gambar diagramnya :



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

```
public class NamaClassPengguna extends Superclass
                    implements InterfaceKe1, InterfaceKe2,
                        InterfaceKe3
{
  // beberapa kode di sini
}
```

Gambar diagramnya :



c) Pewarisan Antar Interface

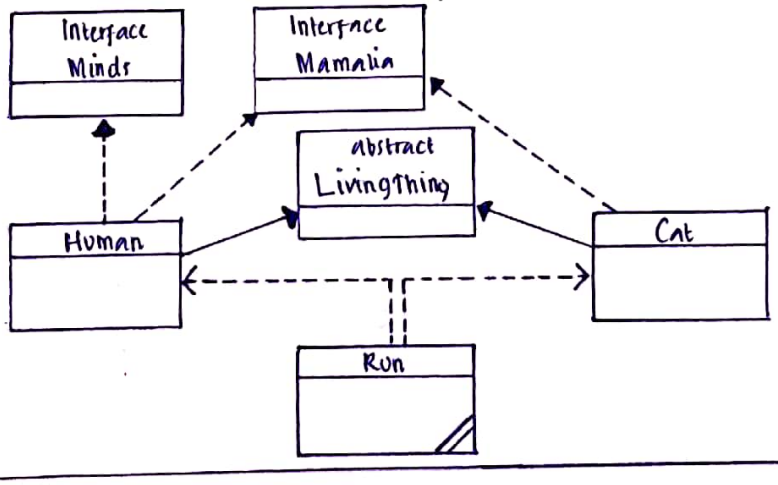
Interface bukan bagian dari hirarki class. Bagaimanapun,

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 mewariri semua deklarasi method dalam PersonInterface.

```
public interface PersonInterface
{
  // beberapa code
}
public interface StudentInterface extends PersonInterface
{
  // beberapa code
}
```

2.2 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, namun 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 abstract class

- Abstract class LivingThing

2 interface

- Interface Mamalia
- Interface Minds

1 class untuk menjalankan yaitu

- Class run

- Membuat abstract class LivingThing :

```
public abstract class LivingThing
{
  // method biasa
  public void breath() {
    System.out.println("Makhluk hidup pasti bernafas...");
  }
  // method biasa
  public void eat() {
    System.out.println("Makhluk hidup pasti Makan...");
  }
  /**
   * Abstract method walk
   * Kita ingin method ini di-overmelden oleh subclass
   */
  public abstract void walk();
}
```

Method walk merupakan method abstract, ketika class meng-extend class abstract LivingThing, diboluhkan untuk override method abstract walk(),

- Membuat Interface Mamalia :

```
public interface Mamalia
{
```

```
        // Method tanpa tubuh
        public void characteristic();
    }
```

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

- Membuat interface Minds :

```
public interface Minds
{
    // Method tanpa tubuh
    public void mind();
}
```

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

- Membuat Class Human :

```
public class Human extends LivingThing implements Mamalia, Minds
{
    // Membuat method
    public void name() {
        System.out.println("Manusia");
    }
    // mengoverride method dari LivingThing
    public void walk() {
        System.out.println("Manusia berjalan menggunakan dua kaki");
    }
    // mengoverride method dari interface mamalia
    public void characteristic() {
        System.out.println("Manusia melahirkan dan menyusui");
    }
    // mengoverride method dari interface minds
    public void mind() {
        System.out.println("Manusia memiliki Akal Pikiran");
    }
}
```

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

```
public class Cat extends LivingThing implements Mamalia
{
    //membuat method
    public void name() {
        System.out.println("Kucing");
    }
    //mengoverride method
    public void walk() {
        System.out.println("Kucing berjalan menggunakan empat kaki");
    }
    //mengOveride method dari interface mamalia
    public void characteristic() {
        System.out.println("Kucing melahirkan dan menyusui");
    }
}
```

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 characteristic dari interface Mamalia

- Membuat class Run yang digunakan untuk menjalankan program

```
public class Run
{
    public static void main() {
    //membuat Objek human dari class Human
    Human human = new Human();
    //membuat objek kucing dari class kucing
    Cat cat = new Cat();
    human.name();
```

```
        human.eat();
        human.breath();
        human.walk();
        human.characteristic();
        human.mind();
        cat.name();
        cat.eat();
        cat.breath();
        cat.walk();
        cat.characteristic();
        }
    }
```

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

2.3 Running Program

```
Manusia
Makhluk hidup pasti Makan...
Makhluk hidup pasti bernafas...
Manusia berjalan menggunakan dua kaki
Manusia melahirkan dan menyusui
Manusia memiliki Akal pikiran
Kucing
Makhluk hidup pasti Makan
Makhluk hidup pasti bernafas
Kucing berjalan menggunakan empat kaki
Kucing melahirkan dan menyusui
```

BAB III

TUGAS PENDAHULUAN

3.1 Soal

Jelaskan perbedaan interface dan polimorfisme!

3.2 Jawab

- Interface

  Interface adalah sebuah kontrak atau spesifikasi yang mendefinisikan metode - metode yang harus ada dalam kelas yang mengimplementasikannya. Interface hanya mendefinisikan tanda - tanda metode (nama, parameter, tipe kembalian), tetapi tidak menyediakan implementasi konkret dari metode - metode tersebut.

- Polimorfisme

  Polimorfisme adalah kemampuan objek untuk mengambil banyak bentuk. Hal ini memungkinkan objek untuk diproses dengan cara yang berbeda tergantung pada tipe yang digunakan untuk merujuk objek tersebut.

- Perbedaan

  Perbedaan utamanya yaitu Interface adalah kontrak yang mendefinisikan tanda - tanda metode, sedangkan polimorfisme adalah konsep yang mengizinkan objek untuk mengambil banyak bentuk.

BAB IV

IMPLEMENTASI

4.1 Soal

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

4.2 Jawaban

a) Hasil

```
I N T E R F A C E
-------------
PC Hidup!
PC Mati
PC Telah diKlik Kanan
PC Telah Diklik Kiri
PC Enter
PC Mencetak Data
--------------
Laptop Hidup!
Laptop Mati
Laptop Telah Diklik Kanan
Laptop Telah Diklik Kiri
Laptop Enter
Laptop Mencetak Data
---------------
Netbook Hidup!
Netbook Mati
Netbook Telah Diklik Kanan
Netbook Telah Diklik Kiri
Netbook Enter
Netbook Mencetak Data
----------------
BUILD SUCCESSFUL (total time : 0 seconds)
```

b) Source Code

1) Komputer

```
package Project6;
Abstract class Komputer implements Mouse, Keyboard, Printer {
    abstract void hidupkan_os();
    abstract void matikan_os();
}
```

2) Mouse

```
package Project6;
public interface Mouse {
    public void klik_kanan();
    public void klik_kiri();
}
```

3) Keyboard

```
package Project6;
public interface Keyboard {
    public void tekan_enter();
}
```

4) Printer

```
package Project6;
public interface Printer {
    public void cetak_data();
}
```

5) PC

```
package Project6;
public class PC extends Komputer implements Mouse, Keyboard, Printer {
    @Override
    public void hidupkan_os() {
        System.out.println("INTERFACE");
```

```
        System.out.println("------------------");
        System.out.println("PC Hidup!");
    }
    @Override
    public void matikan_os(){
      System.out.println("PC Mati");
    }
    @Override
    public void klik_kanan(){
      System.out.println("PC Telah Diklik Kanan");
    }
    @Override
    public void klik_kiri(){
      System.out.println("PC Telah Diklik Kiri");
    }
    @Override
    public void tekan_enter(){
      System.out.println("PC Enter");
    }
    @Override
    public void cetak_data(){
      System.out.println("PC Mencetak Data");
    }
}
```

c) Laptop

```
package Project6;

public class Laptop extends Komputer implements Mouse, Keyboard,
Printer {
  @Override
  public void hidupkan_os(){
    System.out.println("Laptop Hidup!");
  }
```

```
    @Override
    public void matikan_os(){
      System.out.println("Laptop Mati");
    }
    @Override
    public void klik_kanan() {
      System.out.println("Laptop Telah Diklik Kanan");
    }
    @Override
    public void klik_kiri() {
      System.out.println("Laptop Di Klik Kiri");
    }
    @Override
    public void tekan_enter() {
       System.out.println("Laptop Enter");
    }
    @Override
    public void cetak_data() {
      System.out.println("Laptop Mencetak Data");
    }
  }
```

7) Netbook

```
  package Project6;
  public class Netbook extends Komputer implements Mouse, Keyboard,
  Printer {
    @Override
    public void hidupkan_os() {
      System.out.println("Netbook Hidup!");
    }
    @Override
    public void matikan_os() {
      System.out.println("Netbook Mati");
    }
```

```
    @Override
    public void Klik_kanan() {
      System.out.println("Netbook Telah Diklik Kanan");
    }
    @Override
    public void Klik_kiri() {
      System.out.println("Netbook Telah Diklik Kiri");
    }
    @Override
    public void tekan_enter() {
      System.out.println("Netbook Enter");
    }
    @Override
    public void cetak_data() {
      System.out.println("Netbook Mencetak Data");
    }
}
```

8) KomputerCetak

```
package Project6;
public final class KomputerCetak {

  private static void Cetak(Komputer[] obj) {
    obj[0] = new PC();
    obj[1] = new Laptop();
    obj[2] = new Netbook();

    for(int i=0; i < obj.length; i++) {
      obj[i].hidupkan();
      obj[i].matikan();
      obj[i].Klik_kanan();
      obj[i].Klik_kiri();
      obj[i].tekan_enter();
      obj[i].cetak_data();
    }
  }
```

```
    public static void main (String[] args) {
        Komputer [] A = new Komputer [3];
        Cetak (A);
    }
}
```

c) Penjelasan

Pada pembuatan program menggunakan konsep interface ini memerlukan 8 file yang terdiri dari 1 abstract class (Komputer), 3 file class interface (Mouse, Keyboard, Printer), 3 file class (PC, Laptop, Netbook), dan 1 file main class. Ke-8 file ini tentunya saling terhubung dengan perintah extends (abstract class) dan implements (class interface). Kemudian semua methodnya diisi sesuai dengan nama methodnya dan akan dicetak pada final classnya.

BAB V
PENUTUP

5.1 Analisa

Dari hasil praktikum, praktikan menganalisa bahwa penggunaan interface dalam sebuah program memiliki peran penting dalam memfasilitasi interaksi antara pengguna dengan sistem komputer. Interface, atau antarmuka adalah kontak antara pengguna dan program yang memungkinkan pengguna berinteraksi dengan program tersebut. Penggunaan interface penting untuk mengutamakan pengalaman pengguna, efisiensi penggunaan, keseragaman, dan dukungan untuk diversitas pengguna. Dengan mempertimbangkan faktor-faktor ini, pengembang dapat menciptakan antarmuka yang efektif, mudah dipelajari, dan dapat meningkatkan produktivitas dan kepuasan pengguna.

5.2 Kesimpulan

1. Interface (antarmuka) merupakan suatu mekanisme dalam Java yang memungkinkan untuk berbagi konstanta atau menentukan bentuk method yang dapat digunakan oleh sejumlah class.
2. Penggunaan interface yang baik dalam sebuah program memberikan pengalaman pengguna yang baik, meningkatkan efisiensi penggunaan, dan mempertimbangkan kebutuhan diversitas pengguna.
3. Pada program yang telah diimplementasikan di atas menggunakan konsep interface dan beberapa perintah lainnya.