7️⃣

Modul 7 - Abstract Class & interface

Durasi Modul = 2-5 menit

Capaian pembelajaran:

- Memahami konsep dan penggunaan abstract class dalam Java
- Menjelaskan perbedaan antara abstract class dan interface
- Mengimplementasikan interface dalam program Java
- Menerapkan prinsip abstraksi menggunakan abstract class dan interface
- Menganalisis situasi yang tepat untuk menggunakan abstract class atau interface dalam desain program

Deskripsi Pembelajaran

Modul ini akan membahas konsep dasar dan penerapan abstract class dan interface dalam pemrograman Java. Materi yang akan dibahas meliputi:

- Pengertian dan karakteristik abstract class
- Cara membuat dan menggunakan abstract class
- Konsep dan tujuan interface dalam Java
- Implementasi interface pada class
- Perbedaan antara abstract class dan interface
- Contoh kasus penggunaan abstract class dan interface dalam desain program

Melalui modul ini, mahasiswa akan belajar bagaimana memanfaatkan abstract class dan interface untuk meningkatkan abstraksi, fleksibilitas, dan struktur

dalam pengembangan perangkat lunak berorientasi objek. Materi akan disajikan melalui penjelasan konsep, contoh kode, dan latihan praktis untuk memperkuat pemahaman.

7.1. Abstract Class

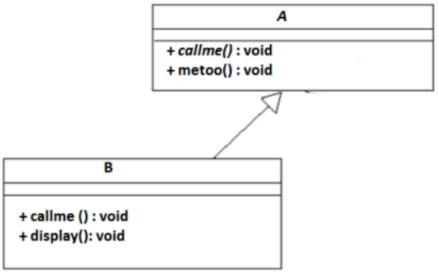
Pada saat melakukan desain menggunakan inheritance, kita meletakkan kode yang bersifat umum kedalam sebuah class yang disebut dengan superclass dan menurunkannya kepada class yang lebih spesifik (subclass). Terkadang superclass menjadi sangat abstract dan tidak dapat membentuk instancenya. Class seperti ini disebut dengan abstract class. Abstract class merupakan class yang terletak pada posisi tertinggi pada hierarki kelas. Class-class yang bersifat abstract sama dengan class-class pada umumnya memiliki atribut dan method. Namun kita tidak dapat membuat instance dari class ini menggunakan operator new, karena suatu objek hanya dapat dibuat dari non-abstract class.

Tujuan kelas abstrak adalah memberikan superclass yang sesuai untuk subclass serta dapat berbagi desain umum. Abstract class memiliki minimal satu method yang bersifat abstrak (abstract method). Abstract method adalah abstract yang hanya berupa deklarasi dan parameter method saja.

▼ **Ciri-Ciri abstract method adalah sebagai berikut:**

1. **Menggunakan keyword abstract**
2. **Tidak memiliki 'tubuh/implementasi' method**

> **Notasi Abstract Class pada Diagram Class**
> Jika
> ***abstract class*** digambarkan menggunakan class diagram akan terlihat seperti gambar dibawah.



> ***Non abstract class*** dihubungkan dengan ***abstract class*** melalui relasi *inheritance* (pewarisan). ***Class yang bersifat abstract*** akan menjadi ***superclass dan nama class*** tersebut ditulis miring.

> **Cara menuliskan abstract class seperti berikut**

```
abstract class nama_class {
        Deklarasi variabel;
        Constructor ()
        Method biasa()
        Method abstract()
}
```

Contoh:
Jika class diagram pada gambar 4.5 diimplementasikan, maka hasilnya seperti berikut.

Class A.java

```
public abstract class A{
 public abstract void callme();
 void metoo()
   { system.out.println ("Insida A's metoo
method");
   }
}
```

Class B.java

```
public abstract class B extends A{
 public void callme(){
 system.out.println ("Call Me from B");
}
}
```

Jika ***subclass*** yang diturunkan dari abstract class tidak mengimplementasikan isi semua method ***abstrak superclass***, maka subclass tersebut harus tetap dideklarasikan ***abstract.*** Deklarasi method abstract pada ***subclass*** tersebut boleh tidak dituliskan kembali. Contoh:

```
public abstract class B extends A{
 public void callme();
```

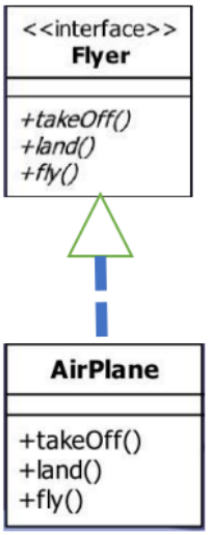
7.2. Interface

Pada saat mengembangkan program kadang kita membutuhkan sebuah class yang mewarisi beberapa *class* sekaligus ***(multiple inheritance).*** Pada beberapa bahasa pemrograman berorientasi objek memungkin kan untuk

mewarisi sifat dari beberapa superclass (lebih dari satu). Namun pada Java hal ini tidak mungkin dilakukan, karena Java hanya mengizinkan subclass mewarisi satu superclass. Untuk mendapatkan efek multiple inheritance di Java, kita dapat menggunakan interface. Interface adalah sebuah blok yang berisi deklarasi metode saja untuk diimplementasikan pada class lain.

Dengan kata lain ***interface*** merupakan prototype kelas yang berisi definisi konstanta dan deklarasi method (hanya nama method tanpa definisi kode programnya). Interface hanya berisi kumpulan konstanta dan method yang tidak memiliki implementasi. Suatu class yang mengimplementasi suatu interface, memiliki "kewajiban" untuk membuat implementasi method-method yang disediakan oleh interface tersebut. Konstanta yang dideklarasikan di dalam interface dapat diwariskan ke class yang mengimplement interface tersebut.

> **Notasi Interface pada Diagram Class**
> Jika interface digambarkan menggunakan class diagram akan terlihat seperti gambar 4.6. Class yang mengimplementasikan interface dihubungkan dengan interface melalui relasi inheritance (pewarisan), namun dengan menggunakan garis panah putus-putus.



Cara untuk mendeklarasikan interface adalah sebagai berikut.

```
public interface nama_interface {
//Deklarasi konstanta;
//Deklarasi abstract method
}
```

Sedangkan untuk memanggil interface dari sebuah class seperti berikut.

```
<modifier> nama_class implements nama_interface{
}
```

Contoh:

Interface Radio.java

```
public interface Radio {
    public void setGelombang (String gelombang);
}
```

Interface Kamera.java

```
public interface Kamera {
    public void setPixel(float pixel);
    public void ambilGambar();

}
```

Class Handphone.java

```
public class Handphone implements Kamera, Radio{
    private String gelombang;
    private float pixel;

    @Override
    public void setPixel(float pixel)
    {
        this.pixel=pixel;
    }
    @Override
    public void ambilGambar()
    {
        System.out.println("Gambar terambil");
    }
    @Override
    public void setGelombang (String gelombang)
    {
    this.gelombang=gelombang;
    }
    @Override
   public String toString()
   {
       return "Handphone dengan kamera : "+pixel+"
pixel dan gelombang radio : "+gelombang;
   }
}
```

📚REFERENSI

Aziefah, & Nurmasari, J. (2013). Pemrograman Berorientasi Objek. Pekanbaru: Politeknik Caltex Riau.

Barclay, K., & Savage, J. (2004). Object Oriented Design with UML and Java. Elsevier.

Deitel, P., & Deitel, H. (2012). Java: How to Program (9 ed.). US: Prentice Hall.