

Implement

Setelah kita mempelajari cara dan ketentuan dalam membuat interface, berikutnya kita akan mempelajari cara menggunakan interface tersebut. Interface harus dapat digunakan ke kelas lain. Cara untuk mengimplementasi interface yaitu menggunakan keyword `implements` pada kelas yang mengimplementasikannya. Kelas yang mengimplementasi interface (bisa lebih dari 1 interface) harus mendefinisikan isi kode semua deklarasi metode yang ada pada interface tersebut.

Codelab Implement

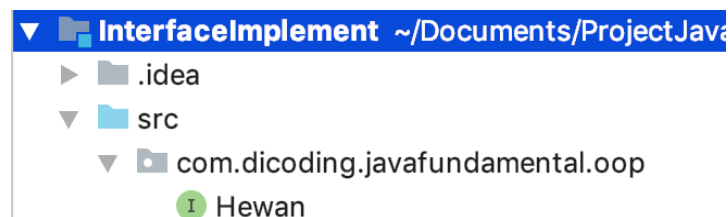
Masih ingat kan kalau metode di interface hanya berupa deklarasi? Jika suatu kelas tak mendefinisikan isi kode semua deklarasi metode yang ada di interface, maka kelas tersebut harus menjadi abstract class. Untuk lebih jelasnya mari kita langsung coding saja.

1. Buatlah proyek baru dengan nama `InterfaceImplement` dengan nama package

`com.dicoding.javafundamental.oop` di dalamnya:



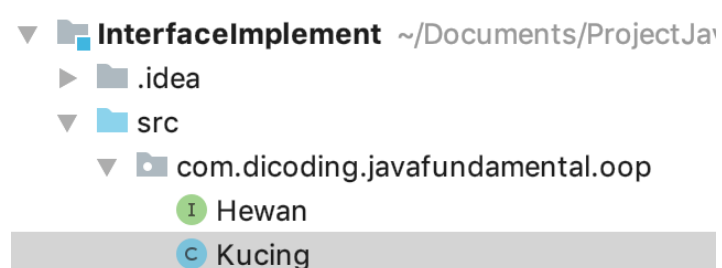
2. Buat interface di package yang baru saja Anda buat. Klik kanan di package tersebut. Pilih New - Java Class. Name: `Hewan`, Kind: Interface, lalu tekan OK.



Tambahkan kode berikut di dalamnya:

```
1. package com.dicoding.javafundamental.oop;
2.
3. public interface Hewan {
4.     String respirasi = "oksigen";
5.
6.     void makan();
7. }
```

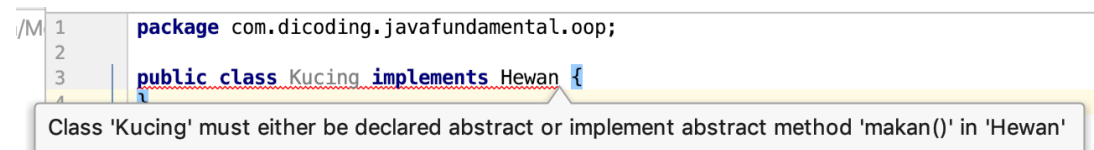
3. Buatlah sebuah kelas baru dengan nama `Kucing`.



Kemudian implementasikan dengan kelas `Hewan` seperti ini:

```
1. package com.dicoding.javafundamental.oop;
2.
3. public class Kucing implements Hewan {
4. }
```

Perhatikan IntelliJ akan memberi informasi kesalahan ditandai baris kode berwarna merah. Jika kita arahkan kursor ke baris tersebut maka akan ada informasi “Class ‘Kucing’ must either be declared abstract or implement abstract method ‘makan()’ in ‘Hewan’”. Ini menjelaskan kalau kita harus memilih untuk menjadikan kelas **Kucing** sebagai abstract class atau mendefinisikan isi kode semua deklarasi metode dari interface **Hewan**.



```
/M 1 package com.dicoding.javafundamental.oop;
2
3 public class Kucing implements Hewan {
4 }
```

Class 'Kucing' must either be declared abstract or implement abstract method 'makan()' in 'Hewan'

4. Jika kita ingin menjadikan kelas **Kucing** sebagai abstract class, cukup tambahkan keyword **abstract** sebelum keyword **class** seperti berikut.

```
1. package com.dicoding.javafundamental.oop;
2.
3. public abstract class Kucing implements Hewan {
4. }
```

5. Jika kita ingin mengimplementasikan interface maka kita harus membuat *method signature* (nama, parameter, return) yang sama disertai isi kode program untuk metodenya. Berikut contoh implementasi metode makan.

```
1. package com.dicoding.javafundamental.oop;
2.
3. public class Kucing implements Hewan {
4.     @Override
5.     public void makan() {
6.
7.     }
8. }
```

Code to Interface not Implementation

Dari latihan sebelumnya kita membuat interface **Hewan**. Kemudian kita membuat kelas **Kucing** yang menerapkan interface tersebut. Maka tiap metode yang berada pada interface harus diimplementasikan. Interface terlihat sebagai suatu aturan atau kontrak bagi kelas yang mengimplementasikannya.

Salah satu contoh implementasi interface adalah JDBC API. Java Database Connectivity (JDBC) adalah package (**java.sql** dan **javax.sql**) yang menyediakan akses ke database. Sedangkan Application Programming Interface (API), dari namanya sudah bisa ditebak adalah kumpulan Interface.

Jadi JDBC API adalah kumpulan interface untuk mengakses database sedangkan implementasinya akan dibuat oleh masing-masing merk database misal MySql, SqlServer, Oracle dan lain-lain. Implementasi tersebut biasa disebut JDBC Driver. JDBC Driver akan kita unduh (dalam bentuk jar) tergantung dari database yang digunakan untuk dimasukkan ke classpath.

Silakan buka tautan <https://www.javatpoint.com/example-to-connect-to-the-mysql-database> untuk melihat contoh JDBC API. Perhatikan penggunaan interface `java.sql.Connection` bukan menggunakan kelas `com.mysql.jdbc.Connection` dari JDBC Driver.

Dalam coding Java ada istilah *best practice (design pattern)* “code to interface not implementation”. Salah satu maksudnya adalah seperti contoh JDBC API di paragraf sebelumnya. Contoh lainnya adalah ketika kita membuat aplikasi yang terdiri dari beberapa layer. Maka lebih baik gunakan interface untuk komunikasi antar layernya.

Silakan baca beberapa tautan ini untuk menambah pengetahuan Anda.

- <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>
- https://en.wikipedia.org/wiki/Design_Patterns

[← Sebelumnya](#)

[Selanjutnya →](#)



Dicoding Space
Jl. Batik Kumeli No.50, Sukaluyu,
Kec. Cibeunying Kaler, Kota Bandung
Jawa Barat 40123

Penghargaan



Decode Ideas
Discover Potential

[➤ Tentang Kami](#)

- [Blog](#)
- [Reward](#)
- [Showcase](#)
- [Hubungi Kami](#)
- [FAQ](#)