Interface

Apa itu interface? Di materi sebelumnya kita sudah mempelajari class.

Pada bahasa pemrograman Java interface sangat mirip dengan kelas, tapi tanpa atribut kelas dan memiliki metode yang dideklarasikan tanpa isi. Deklarasi metode pada sebuah interface dapat diimplementasikan oleh kelas lain.

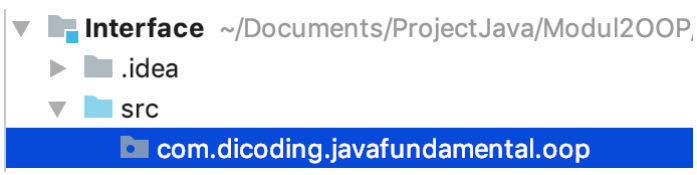
Sebuah kelas dapat mengimplementasikan lebih dari satu interface. Kelas ini akan mendeklarasikan metode pada interface yang dibutuhkan oleh kelas itu sekaligus mendefinisikan isinya pada kode program.

Metode pada interface yang diimplementasikan pada suatu kelas harus sama persis dengan yang ada pada interface tersebut. Property/Field di interface akan menjadi `static final` atau `konstanta`. Method dan field di interface akan selalu bersifat `public`. Perhatikan kata-kata yang diberi huruf tebal.

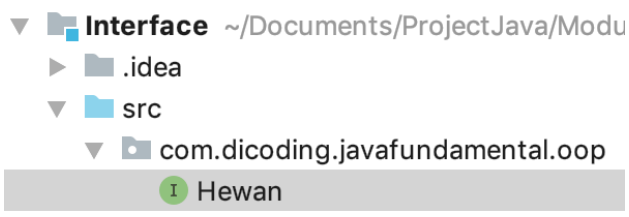
Codelab Interface

Untuk memperjelas implementasinya, mari kita mulai saja:

1. Buatlah proyek baru dengan nama `Interface` dengan nama package `com.dicoding.javafundamental.oop` di dalamnya:



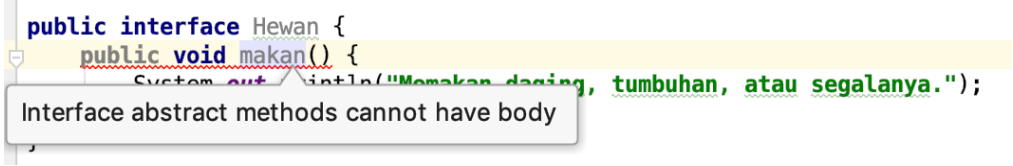
2. Buat `interface` di package yang baru saja Anda buat. Klik `kanan` di package tersebut. Pilih `New - Java Class`. Name: `Hewan`, Kind: `Interface`, lalu tekan OK.



3. Tambahkan baris kode di bawah ini ke interface `Hewan`.

```
1.  package com.dicoding.javafundamental.oop;
2.
3.  public interface Hewan {
4.      public void makan() {
5.          System.out.println("Memakan daging, tumbuhan, atau segalanya.");
6.      }
7.  }
```

IntelliJ memberi tanda eror. Apa yang terjadi? Coba arahkan kursor ke baris yang error (ditandai warna merah) IntelliJ akan memberi informasi “`Interface abstract method cannot have body`”. Yup, ini karena method di interface hanya berupa deklarasi seperti yang sudah dibahas di awal paragraf.



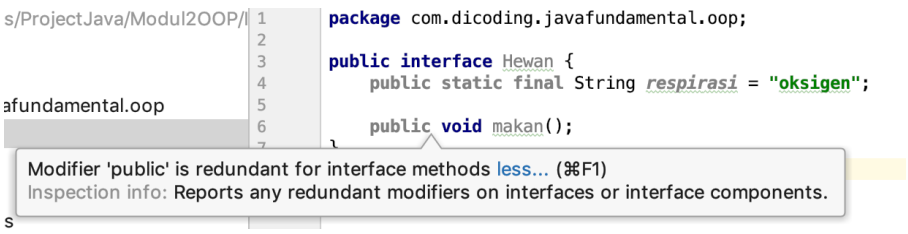
Ubah method makan menjadi deklarasi seperti baris kode di bawah ini.

```
1. package com.dicoding.javafundamental.oop;
2.
3. public interface Hewan {
4.     public void makan();
5. }
```

Tambahkan baris kode di bawah ini ke interface `Hewan`.

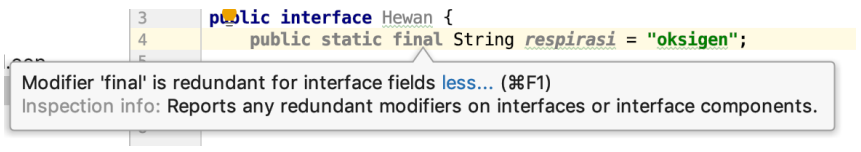
```
1. package com.dicoding.javafundamental.oop;
2.
3. public interface Hewan {
4.     public static final String respirasi = "oksigen";
5.
6.     public void makan();
7. }
```

4. Posisi saat ini tidak ada eror yang dilaporkan IntelliJ, tetapi sebagian kode kita menjadi berwarna abu-abu. Coba arahkan kursor ke bagian kode yang berwarna abu-abu. Pertama ke modifier public di property atau method yang kita buat. Akan ada informasi “Modifier ‘public’ is redundant for interface fields..”. Ini menjelaskan kalimat di paragraf di awal yaitu method dan field akan selalu bersifat public.



5. Selain modifier public, kode yang berwana abu-abu adalah keyword static. Coba arahkan kursor ke bagian tersebut. Akan ada informasi “Modifier ‘static’ is redundant for interface fields..”. Ini menandakan keyword static juga tidak diperlukan karena field di interface akan dianggap sebagai static.

6. Begitu juga untuk keyword final tidak diperlukan. Apabila kursor diarahkan ke bagian keyword final maka akan ada informasi “Modifier ‘final’ is redundant for interface fields..”. Ini menjelaskan kalimat di paragraf awal yaitu property/field akan selalu bersifat static final.



7. Berdasarkan penjelasan sebelumnya maka kita bisa mengubah kode menjadi seperti di bawah ini. Tidak ada eror yang dilaporkan IntelliJ. Perhatikan masih ada kode yang diwarnai abu-abu karena method atau field “... is never used”. Hal ini wajar.

```java
package com.dicoding.javafundamental.oop;

public interface Hewan {
    String respirasi = "oksigen";

    void makan();
}
```

8. Ada satu hal lagi yang perlu dibahas dari penjelasan kalimat pada paragraf di awal yaitu field akan menjadi static final atau konstanta. Konstanta di kode Java biasanya ditulis dengan huruf besar semuanya dan diberi pemisah *underscore* jika lebih dari 1 kata. Hal ini tertuang di java coding convention bagian Constants. Coding convention sangat diperlukan dalam bahasa pemrograman apapun. Gunanya, mempermudah developer membaca kode yang ditulis oleh orang lain atau bahkan diri sendiri. Sehingga interface `Hewan` selengkapnya menjadi seperti di bawah ini.

```java
package com.dicoding.javafundamental.oop;

public interface Hewan {
    String RESPIRASI = "oksigen";

    void makan();
}
```

Baca tautan ini untuk mengerti coding convention lebih lanjut Code Conventions.

Sebelumnya

Selanjutnya