**Implicit Interface**

Selain *abstract class*, cara lain yang bisa kita gunakan untuk menerapkan abstraksi dalam OOP adalah dengan ***interface***. *Interface* atau antarmuka merupakan set instruksi yang bisa diimplementasi oleh objek. Secara umum, *interface* berfungsi sebagai penghubung antara sesuatu yang abstrak dengan sesuatu yang nyata.

Bayangkan *remote TV* atau tombol yang ada di HP Anda. Tombol-tombol ini bisa kita sebut sebagai *interface*. Kita tak perlu tahu dan peduli tentang bagaimana fungsi yang ada di dalamnya.

Dart tidak memiliki *keyword* atau *syntax* untuk mendeklarasikan *interface* seperti bahasa pemrograman OOP lainnya. Setiap *class* di dalam Dart dapat bertindak sebagai *interface*. Maka dari itu *interface* pada Dart dikenal sebagai ***implicit interface***. Untuk mengimplementasikan *interface*, gunakan *keyword* implements. Kita bisa mengimplementasikan beberapa *interface* sekaligus pada satu kelas.

1. class ClassName implements InterfaceName

Setelah kelas mengimplementasikan *interface*, maka kelas tersebut wajib mengimplementasikan semua metode yang ada di dalam *interface*. Misalnya kita buat kelas baru bernama Flyable yang akan bertindak sebagai *interface*.

1. class Flyable {
2. void fly() { }
3. }

Kita dapat membiarkan *body* dari *method* fly() tetap kosong karena fungsi implementasinya akan dilakukan oleh class. Selanjutnya buat kelas baru yang mengimplementasi *interface* Flyable.

1. class Bird extends Animal implements Flyable {
2. String featherColor;
4. Bird(String name, int age, double weight, this.featherColor) : super(name, age, weight);
6. }

Anda akan mendapati eror yang memberikan pesan “*Missing concrete implementation of Flyable.fly*”. Ini artinya kita harus mengimplementasi fungsi fly yang terdapat pada *interface* Flyable.

1. class Bird extends Animal implements Flyable {
2. String featherColor;
4. Bird(String name, int age, double weight, this.featherColor) : super(name, age, weight);
6. @override
7. void fly() {
8. print('$name is flying');
9. }
11. }

*Keyword* atau anotasi @override menunjukkan fungsi tersebut mengesampingkan fungsi yang ada di *interface* atau kelas induknya, lalu menggunakan fungsi yang ada dalam kelas itu sendiri sebagai gantinya.