**Inheritance**

Beberapa objek bisa memiliki beberapa karakteristik atau perilaku yang sama, namun sebenarnya mereka bukanlah objek yang sama. Di sini hadirlah peran *inheritance* atau pewarisan. Apa definisi keduanya? ***Inheritance*** adalah kemampuan suatu program untuk membuat kelas baru dari kelas yang ada. Konsep *inheritance* ini bisa dibayangkan layaknya seorang anak mewarisi sifat dari orang tuanya. Di dalam OOP kelas yang menurunkan sifat disebut sebagai kelas induk (*parent class/superclass*) sementara kelas yang mewarisi kelas induknya disebut sebagai kelas anak (*child class/subclass*).

Yuk kembali lagi pada contoh objek kucing. Selain kucing ada jenis hewan lain yang bersifat sama. Misalnya ikan dan burung juga memiliki nama, berat, dan umur. Selain itu mereka juga melakukan aktivitas seperti makan dan tidur. Yang membedakan objek tersebut adalah cara mereka bernafas dan bergerak. Untuk lebih memahami, perhatikanlah tabel kelas berikut:

|  |  |  |
| --- | --- | --- |
| **Cat** | **Fish** | **Bird** |
| **+ name + weight + age** + furColor | **+ name + weight + age** + skinColor | **+ name + weight + age** + featherColor |
| **- eat() - sleep() - poop()** - walk() | **- eat() - sleep() - poop()** - swim() | **- eat() - sleep() - poop()** - fly() |

Bisa kita lihat pada tabel di atas bahwa objek Cat, Fish, dan Bird memiliki beberapa *property* dan *method* yang sama seperti name, weight, age, eat(), dan sleep().

Dibandingkan membuat 3 kelas dan menuliskan ulang properti yang sama, kita bisa memanfaatkan teknik *inheritance* dengan mengelompokkan properti dan fungsi yang sama. Caranya buat sebuah kelas baru yang nantinya akan diturunkan sifatnya.

|  |
| --- |
| **Animal** |
| + name + weight +age |
| - eat() - sleep() - poop() |

|  |  |  |
| --- | --- | --- |
| **Cat** | **Fish** | **Bird** |
| + furColor | + skinColor | + featherColor |
| - walk() | - swim() | - fly() |

Setelah membuat kelas Animal, kita dapat membuat kelas lainnya lalu melakukan ***extends*** ke kelas induknya. Untuk menerapkan *inheritance* gunakan *keyword* extends seperti contoh berikut:

1. class ChildClass extends ParentClass {
3. }

Dengan begitu kita bisa membuat kelas Cat mewarisi kelas Animal.

* [**Cat.dart**](https://www.dicoding.com/academies/191/tutorials/7618?from=16298#tab1-code1)
* [Animal.dart](https://www.dicoding.com/academies/191/tutorials/7618?from=16298#tab1-code2)

1. import 'Animal.dart';
3. class Cat extends Animal {
4. String furColor;
6. Cat(String name, int age, double weight, String furColor) : super(name, age, weight) {
7. this.furColor = furColor;
8. }
10. void walk() {
11. print('$name is walking');
12. }
14. }

Karena kelas Cat adalah turunan dari kelas Animal, maka kita bisa mengakses sifat dan perilaku dari Animal melalui kelas Cat.

1. import 'Cat.dart';
3. void main() {
4. var dicodingCat = Cat('Grayson', 2, 2.2, 'Gray');
5. dicodingCat.walk();
6. dicodingCat.eat();
7. print(dicodingCat.weight);
8. }
10. /\*
11. Output :
12. Grayson is walking
13. Grayson is eating.
14. 2.4000000000000004
15. \*/

**Inheritance constructor**

Karena kelas Animal memiliki *constructor* untuk menginisialisasi properti di dalamnya, maka semua kelas turunannya juga perlu mengimplementasikan *constructor* tersebut. Oleh sebab itu ketika membuat kelas Cat tanpa mendefinisikan *constructor* kita akan mendapatkan eror. IntelliJ IDEA akan memberikan saran untuk membuat *constructor*.

1. Cat(String name, int age, double weight) : super(name, age, weight);

*Keyword* super di atas akan diarahkan ke *constructor* dari kelas Animal.

Jika ingin menginisialisasikan nilai furColor melalui *constructor*, maka kita bisa menambahkan parameter di dalam *constructor*.

1. Cat(String name, int age, double weight, String furColor) : super(name, age, weight) {
2. this.furColor = furColor;
3. }

Atau, kita bisa meringkasnya seperti yang telah kita pelajari pada materi *constructor*.

1. Cat(String name, int age, double weight, this.furColor) : super(name, age, weight);