**Java Reflection**

## **Pendahuluan**

**Sebelumnya harus sudah paham materi terkait** :

* Java Dasar
* Java Object Oriented Programming
* Apache Maven
* Java Unit Test

**Agenda**

* Pengenalan Java Reflection
* Class
* Fields
* Method
* Property
* Constructor
* Array
* Annotation
* Parameterized Type
* Dan lain-lain

## **Pengenalan Java Reflection**

* Java Reflection merupakan fitur Java dimana memperbolehkan program Java untuk mempelajari atau memodifikasi dirinya sendiri
* Misal saja kitab isa melihat struktur sebuah Java class dari mulai semua fields, method, constructor dan lain -lain saat aplikasi berjalan
* Bahkan Java Reflection bisa digunakan untuk memodifikasi kode program yang sedang berjalan
* Java Reflection banyak sekali digunakan oleh framework-framework karena memang sangat powerfull sekali

**Package Java Reflection**

* Fitur Java Reflection terdapat dalam dua package, yaitu java.lang dan java.lang.reflect
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/package-summary.html>
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/package-summary.html>
* Ada banyak sekali class yang terdapat dalam Java Reflection, dan akan kita coba baahs satu persatu dalam kelas ini

## **Class**

**Class<T>**

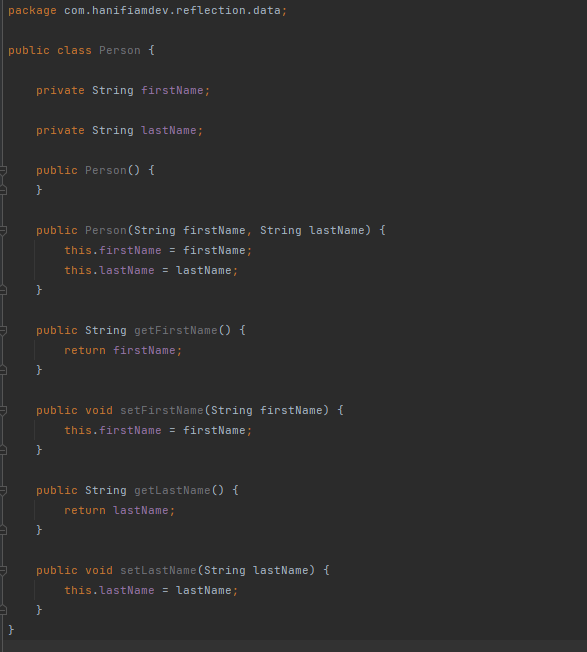
* Java.lang.Class<T> merupakan representasi dari reflection untuk Java Class, Interface dan Enum
* Saat kita membuat Java Class, Interface atau Enum, kadang kita menambahkan field dan method
* Dengan kemampuan java.lang.Class. kita bisa membaca seluruh data member yang terdapat pada Java Class, Interface atau Enum pada saat aplikasi nya berjalan
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Class.html>

**Membuat Class<T>**

* Untuk membuat Class<T>, kitab isa melakukan beberapa cara
* Cara yang pertama dengan meggunakan kata kunci .class setelah nama Java Class, Interface atau Enum nya, misal Person.class, Repository.class, atau Gender.class
* Atau kita juga bisa membuat Class<T> dari sebuah String, menggunakan static method Class.forClass(“com.example.blabla.Person”)
* Atau kita juga bisa mengambil Class<T> dari object, dengan menggunakan method getClass()

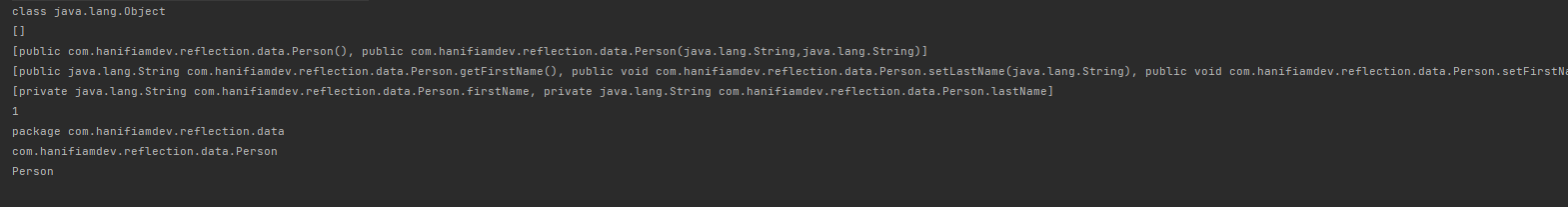
**Class<T> Method**

* Class<T> memiliki banyak sekali method
* Seperti untuk mendapatkan method, fields, constructor, annotation, superclass, interface, dan lain – lian
* Semua detail method nya bis akita baca di Javadoc
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Class.html>



Membuat unit test



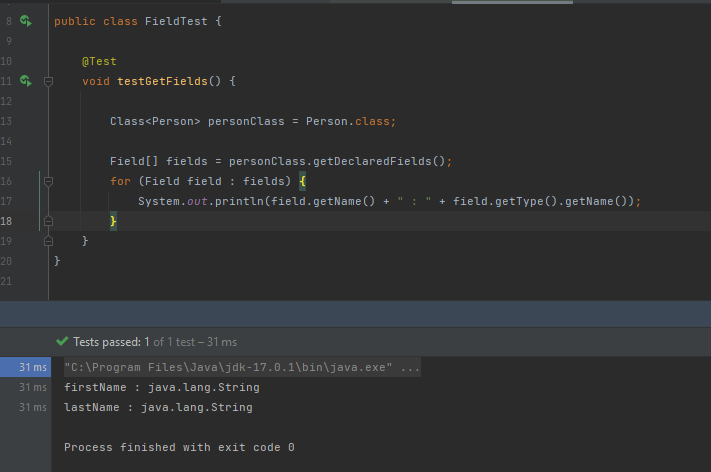


## **Field**

* Field merupakan representasi dari Java Field yang terdapat di dalam Java Class
* Untuk mendapatkan public Field, kita bisa menggunakan method getFields()
* Atau jika ingin mendapatkan semua field dengan semua visibility, kitab isa menggunkan method getDeclaredFields()
* Atau kita juga bisa mendapatkan Field berdasarkan nama field nya menggunakan method

getField(name) atau getDeclaredField(name)

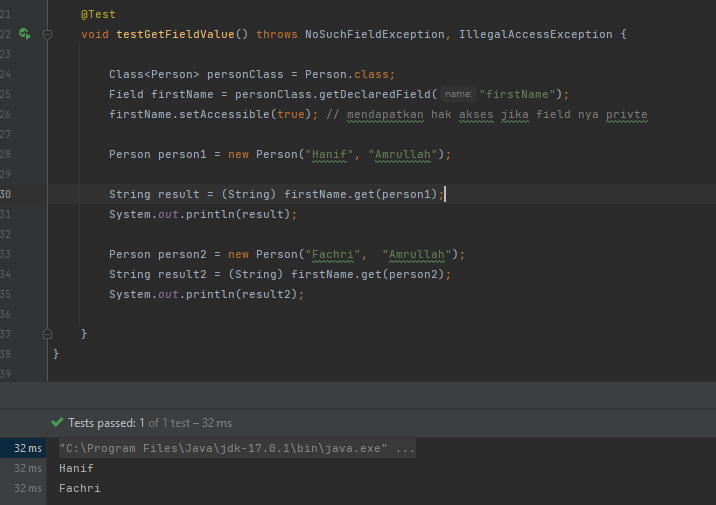
* Field sama seperti Class<T>, memiliki banyak sekali method yang bis akita gunakan untuk melihat detail dari Field tersebut, seperti tipe data, nama field, annotation, dan lain-lain
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/Field.html>



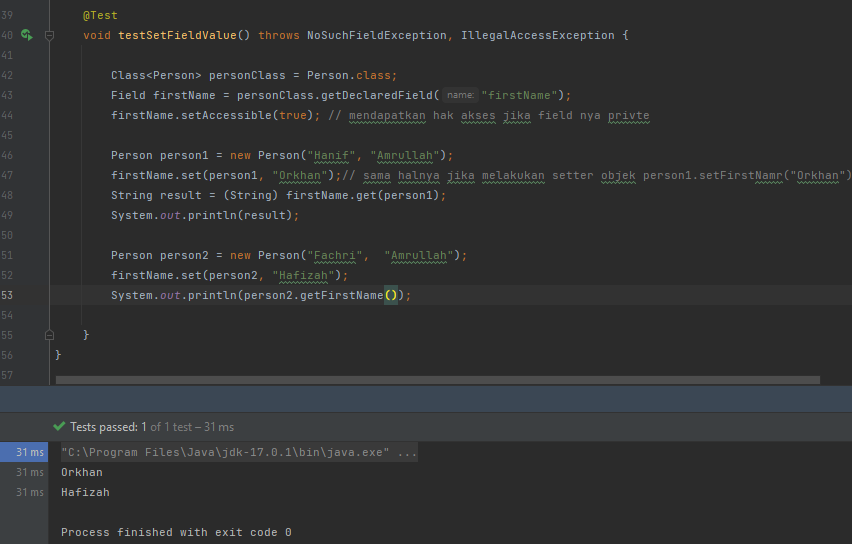
**Mengambil atau Mengubah Field Object**

* Field memiliki kemampuan untuk mengambil atau mengubah field dari object yang ada
* Misal kita sudah membuat Field, lalu kita memiliki object person 1, lalu kita ingin mengambil nilai Field tersebut atau mengubahnya, kita bisa menggunakan method setXxx() atau getXxx() pada Field

Mengambil Field dengan reflection



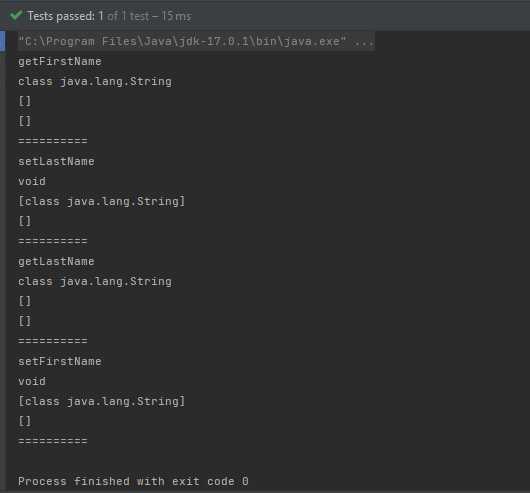
Mengubah Field dengan reflection



## **Method**

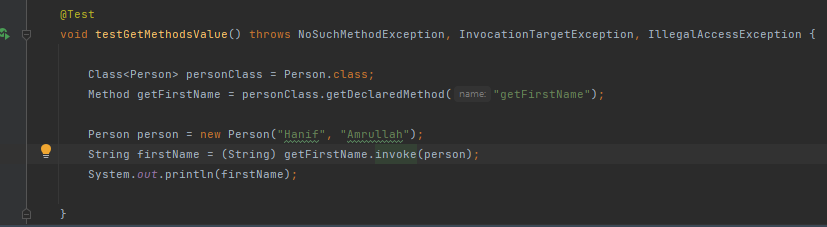
* Selain Field, kita juga bisa mendapatkan Method yang tersedia di Class<T>
* Cara mendapatkan pun sama seperti Field, kita bisa menggunakan method getMethods(), getDeclaredMethods(), getMethod(name), dan getDeclaredMethod(name)
* Method pun banyak sekali method yang bis akita gunakan untuk mendapatkan informasi seperti return value, name, annotation, parameter dan lain-lain
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/Method.html>





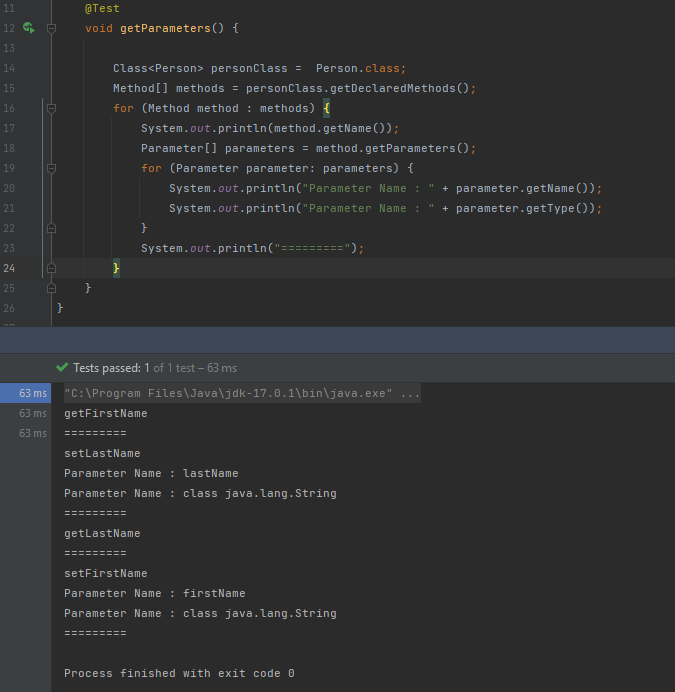
**Memanggil Method Object**

* Method bisa digunakan untuk memanggil method pada sebuah object
* Hampir mirip dengan Field yang bisa digunakan untuk mengambil atau mengubah field didalam object
* Untuk mengambil method object, kita bisa menggunakan method invoke(object, parameters..)



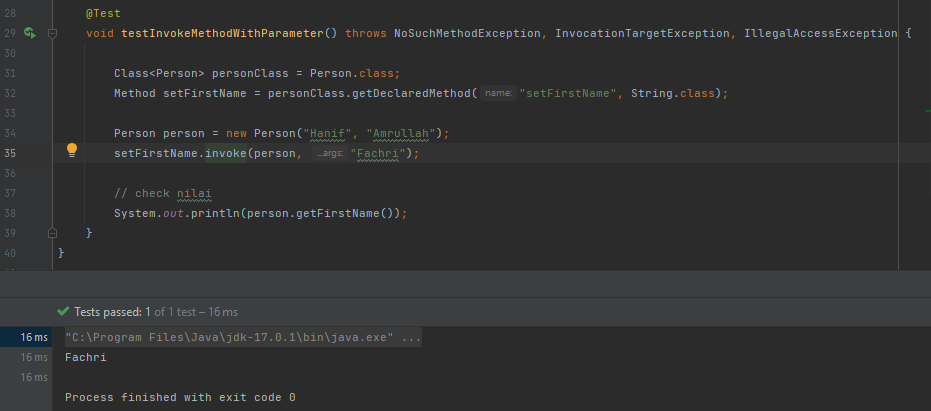
## **Parameter**

* Parameter merupakan representasi dari Java Parameter di Java Method
* Cara mendapatkan Parameter, kita bisa ambil dari Method, karena Parameter memang hanya ada di Method dan Constructor ( yang akan kita bahas nanti )
* Parameter memiliki banyak sekali method, seperti untuk mendapatkan tipe data parameter, nama parameter, dan lain-lain
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/Parameter.html>



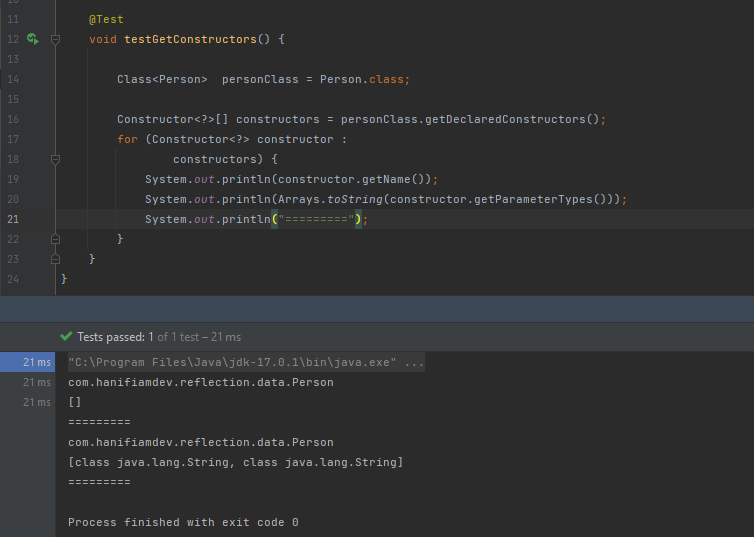
**Memanggil Method Object dengan Parameter**

* Sama seperti Method tanpa parameter
* Kita juga bisa memanggil Method yang memiliki parameter



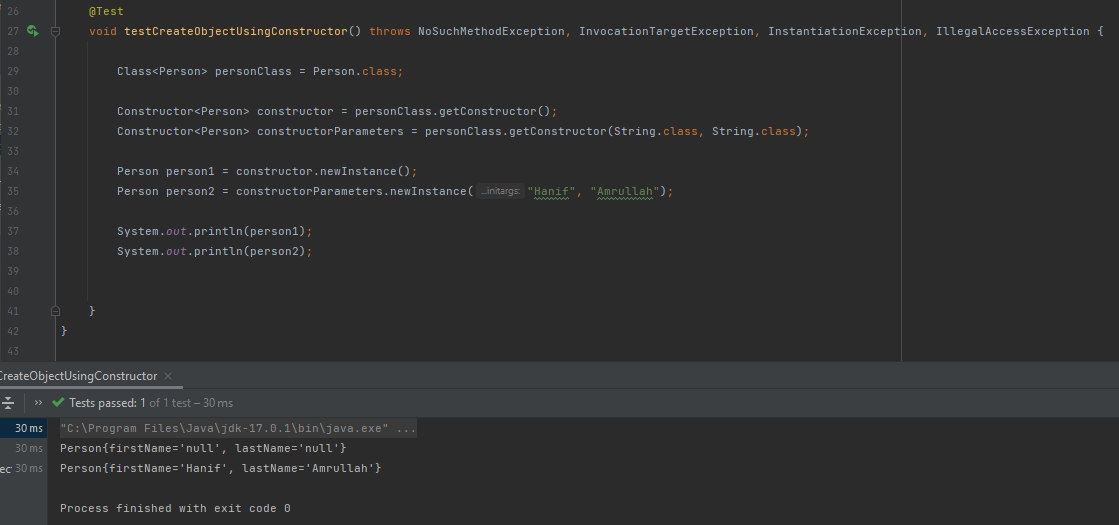
## **Constructor<T>**

* Constructor<T> merupakan representasi dari java Constructor di Java Class
* Constructor<T> ini mirip dengan Method, dimana dia memiliki parameter
* Untuk membuat Constructor kita mendapatkannya melalui Class<T>
* Constructor<T> merupakan tipe data generic, mengikuti tipe data generic dari Class<T> nya
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/Constructor.html>



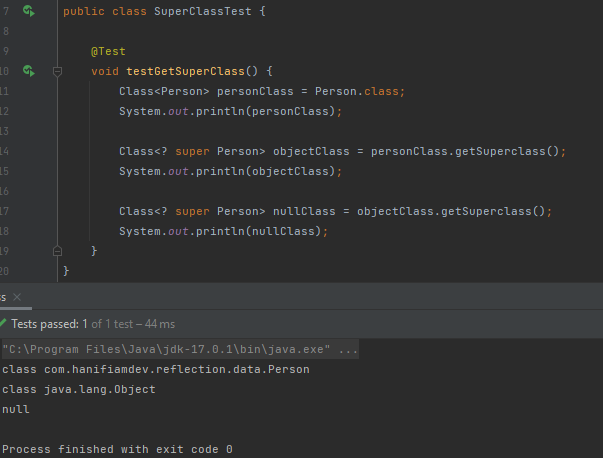
**Membuat Object dengan Constructor**

* Kita sudah tahu bahwa Constructor merupakan method yang dieksekusi ketika sebuah Object pertama kali dibuat
* Dengan menggunakan Constructor, kita juga bisa membuat objet baru
* Caranya dengan menggunakan method newInstance(parameter…)



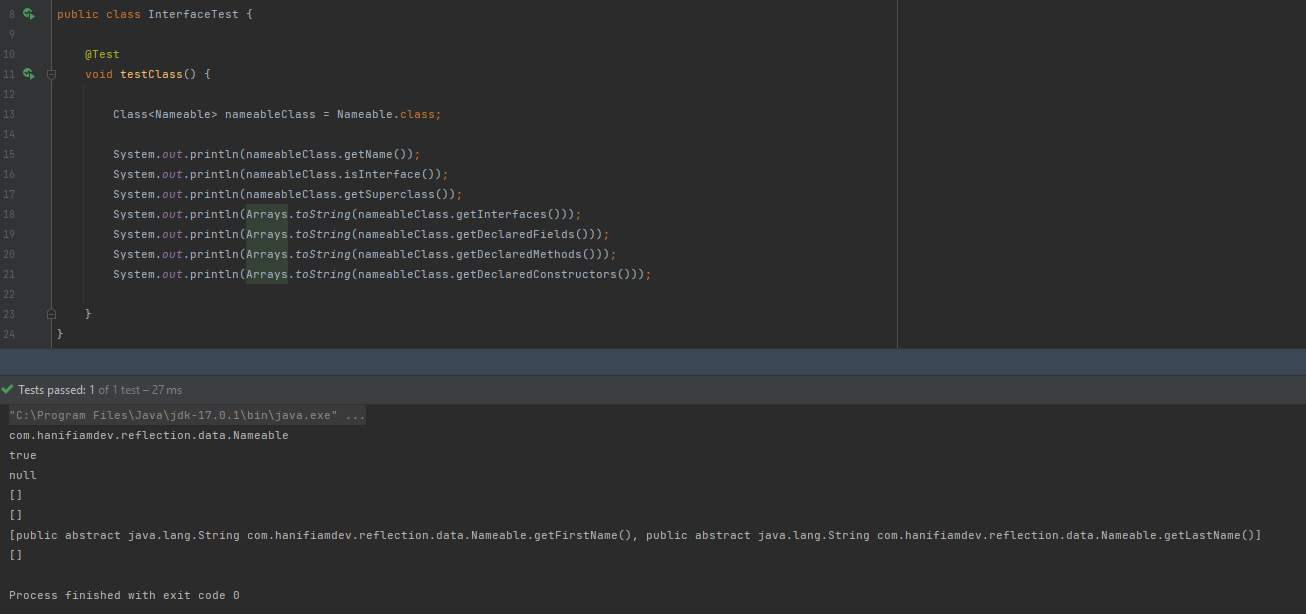
## **Super Class**

* Dengan menggunakan Java Reflection, kita juga bisa mengetahui Super Class dari sebuah Java Class
* Terdapat method getSuperclass() di Class<T> untuk mendapatkan Super Class nya
* Perlu diingat, bahwa saat kita membuat class, jika kita tidak menambahkan super class, secara otomatis super class nya adalah class java.lang.Object
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Class.html#getSuperclass()>



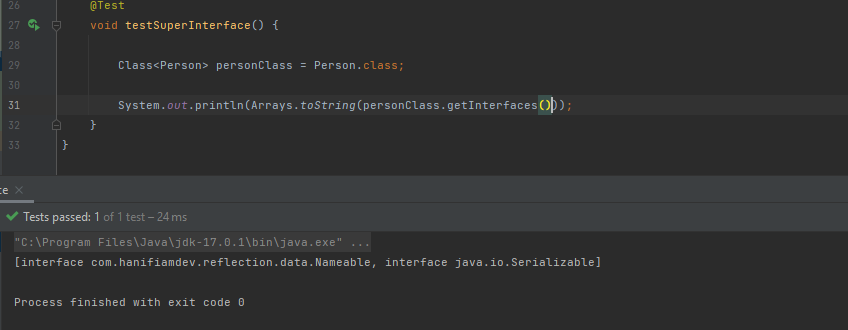
## **Interface**

* Sebelumnya kita hanya bahas tentang Class, bagaimana dengan Interface?
* Interface sebenarnya representasi di Java Reflection tetaplah Class<T>
* Yang membedakan adalah, pada Interface sudah pasti tidak memiliki Constructor dan juga tidak bisa
* Selain itu untuk mengetahui apakah Class<T> ini adalah Java Class atau Java Interface, kita bisa menggunakan method isInterface()



**Super Interface**

* Kita tahu bahwa Interface juga mendukung pewarisan, berbeda dengan class, saat kita melakukan implements Interface, bisa lebih dari satu interface, selain itu interface juga bisa extends Interface
* Pada kasus seperti ini, jika kita melihat semua super interface Class<T>, kita bisa menggunakan method getInterfaces()



## **Modifier**

* Modifier merupakan representasi dari Java Modifier, seperti misal private, protected, public, abstract, dan lain-lain
* Modifier di Java Reflection direpresentasikan oleh number int, dan untuk mempermudah, kita bisa menggunakan class Modifier untuk mengecek jenis modifier nya
* Untuk mendapatkan data modifier, kitab isa menggunakan getModifiers(), entah itu di class, method, field, parameter, constructor, dan lain-lain
* <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/reflect/Modifier.html>

