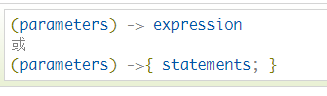
[**Java 8 Lambda 表达式**](https://www.runoob.com/java/java8-lambda-expressions.html)

lambda是什么，一段带有输入参数的可执行语句块。使用lambda表达式可以使我们的代码更加紧凑、简洁。

前置条件：此接口必须是函数式接口（接口内只定义一个方法）。

语法格式：



可选**类型声明**：不需要声明参数类型，编译器可以统一识别参数值。

可选的**参数圆括号**：一个参数无需定义圆括号，但多个参数需要定义圆括号。

可选的**大括号**：如果主体包含了一个语句，就不需要使用大括号。

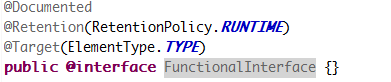
可选的**返回关键字**：如果主体只有一个表达式返回值则编译器会自动返回值，大括号需要指定明表达式返回了一个数值。

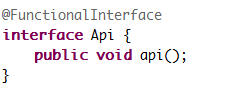
简单实例：



lambda是匿名函数的升级版，所以匿名函数的变量作用域同样适用于lambda表达式。

**@FunctionInterface注解表示该接口是一个函数式接口**





[Java8方法引用（Method Reference）](https://www.cnblogs.com/xiaoxi/p/7099667.html)

方法引用：是用来**直接访问类或者实例的已经存在的方法或者构造方法**。方法引用提供了一种**引用而不执行方法的方式（因为它是lambda表达式）**，它需要由兼容的函数式接口构成的目标类型上下文。计算时，方法引用会**创建函数式接口的一个实例**。

在Java 8中，我们会使用Lambda表达式创建匿名方法，但是有时候，我们的Lambda表达式可能**仅仅调用一个已存在的方法，而不做任何其它事**，对于这种情况，通过一个方法名字来引用这个已存在的方法会更加清晰，Java 8的方法引用允许我们这样做。**方法引用是一个更加紧凑，易读的Lambda表达式**。

注意：方法引用是一个Lambda表达式，其中方法引用的操作符是双冒号"::"。

等效写法，可以看出，使用方法引用代码更加易读、紧凑

// 使用lambda表达式和类的静态方法

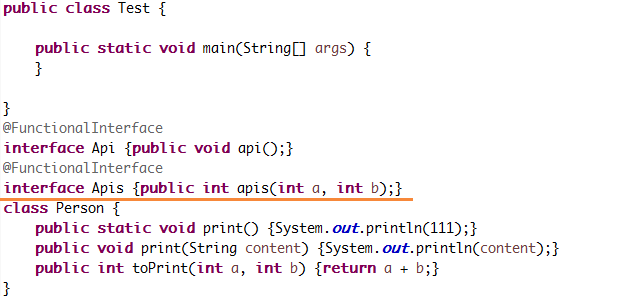
Arrays.sort(pArr, **(a ,b) -> Person.compareByAge(a, b)**);

// 使用方法引用，引用的是类的静态方法

Arrays.sort(pArr, **Person::compareByAge**);

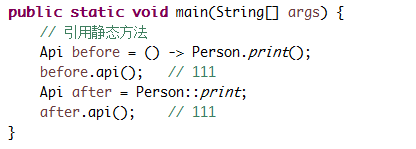
方法引用的四种使用场景

|  |  |
| --- | --- |
| **类型** | **示例** |
| 引用**静态方法** | ContainingClass::staticMethodName，如String::valueOf |
| 引用某个**对象的实例方法**  **a.实例上的实例方法引用**  **b.超类上的实例方法引用** | ContainingObject::instanceMethodName，如  **a. instanceReference::methodName**  b. super**::**methodName或者this::methodName |
| 引用某个类型的任意对象的实例方法 | ContainingType::methodName，如String::compareToIgnoreCase |
| 引用**构造方法**  a.构造方法引用  b.数组构造方法引用 | ClassName::new，如  a. Object::new  b. int[]::new |



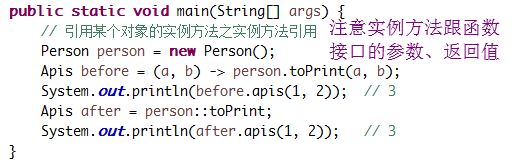
例子：

1、引用静态方法

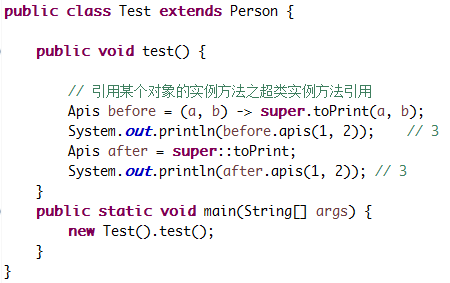
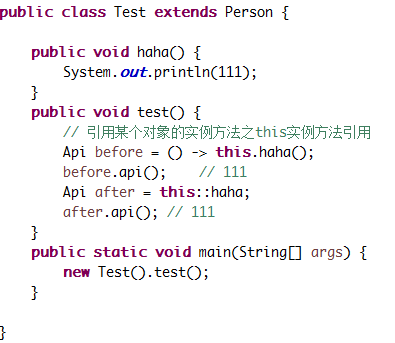


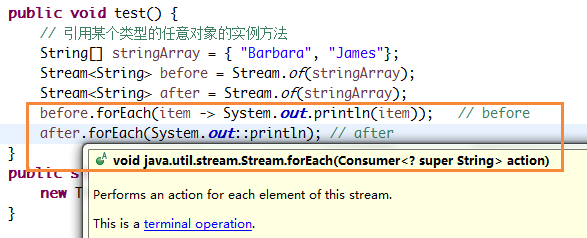
2、引用某个对象的实例方法

2.1 实例上的实例方法引用



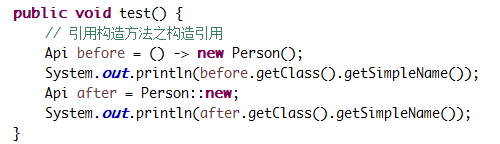
2.2 超类上的实例方法引用

3、引用某个类型的任意对象的实例方法

4、引用构造方法

4.1 构造方法引用



4.2 .数组构造方法引用

