

## 函数的本质

- FunctionalInterface（函数式接口）的本质是**只有一个抽象方法的接口。**

- Lambda转换：

编译器将Lambda表达式转换为私有静态方法（称为"lambda体"）

使用invokedynamic指令指向Lambda的引导方法(bootstrap method)

* **函数式接口访问外部变量的规则：**

1) 局部变量：

- 必须为final或事实上final

- 因为: 捕获的是变量值而非引用

2) 实例变量：

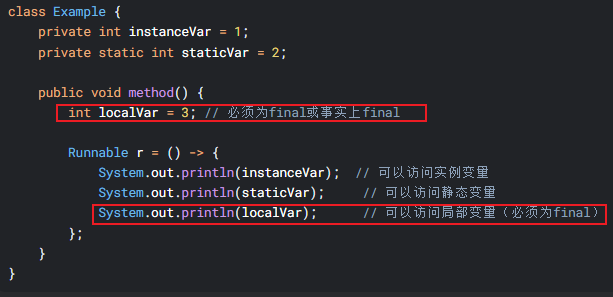
- 可直接访问

- 因为: 实际捕获的是外部类的this引用

3) 静态变量：

- 可直接访问

- 因为: 属于类而非实例



### 规范

Any local variable, formal parameter, or exception parameter used but not declared in a lambda expression must either be final or effectively final ([§4.12.4](https://docs.oracle.com/javase/specs/jls/se17/html/jls-4.html#jls-4.12.4))。

在lambda表达式中使用但**未声明的任何局部变量、形式参数或异常参数**必须是final或有效final（§4.12.4），



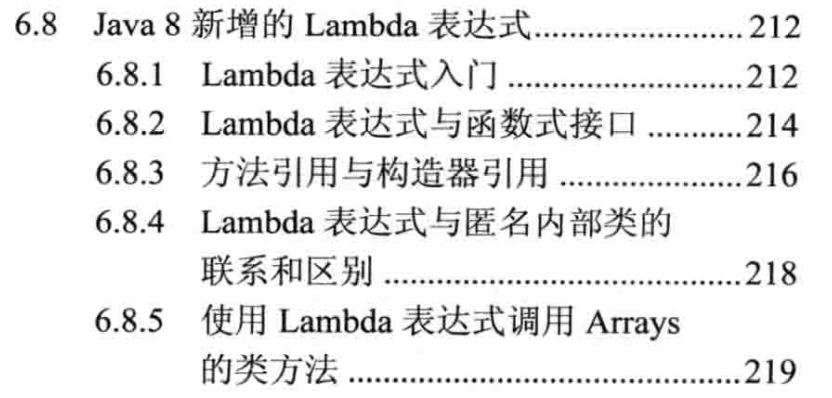
- 与内部类的区别：

\* Lambda表达式使用invokedynamic指令，**不生成.class文件**

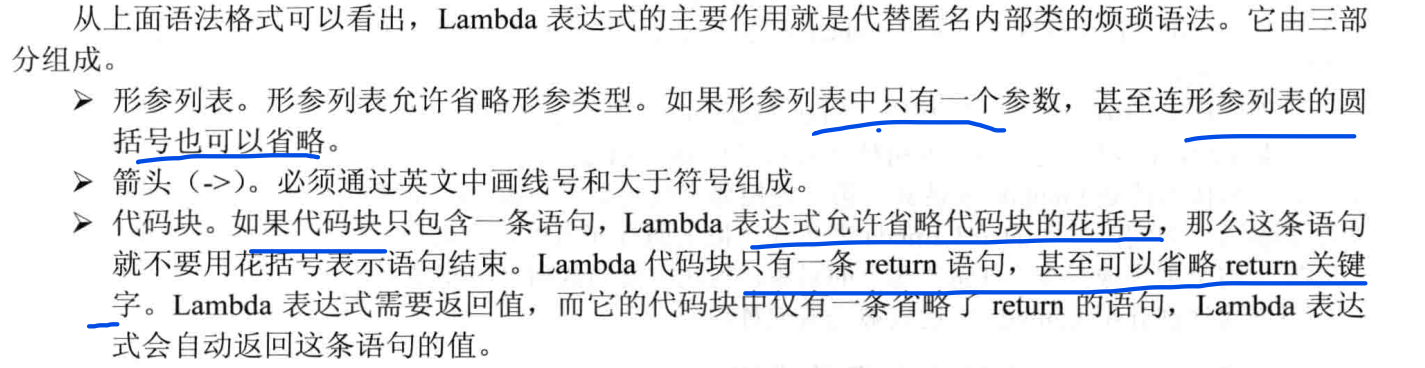
\* 匿名内部类**会生成新.class文件**

\* Lambda表达式性能更高，因为**不需要实例化新对象**

## Lambda表达式



### 语法



无个参数

() -> System.out.println(“111”)

1个参数

Weater ->{

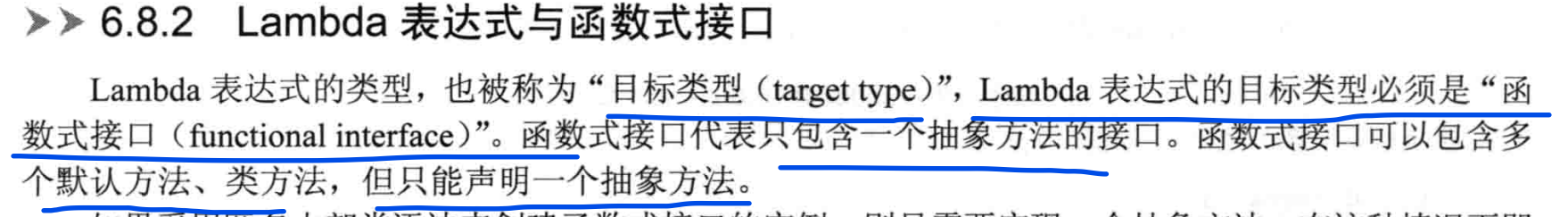
System.out.println(“111”);

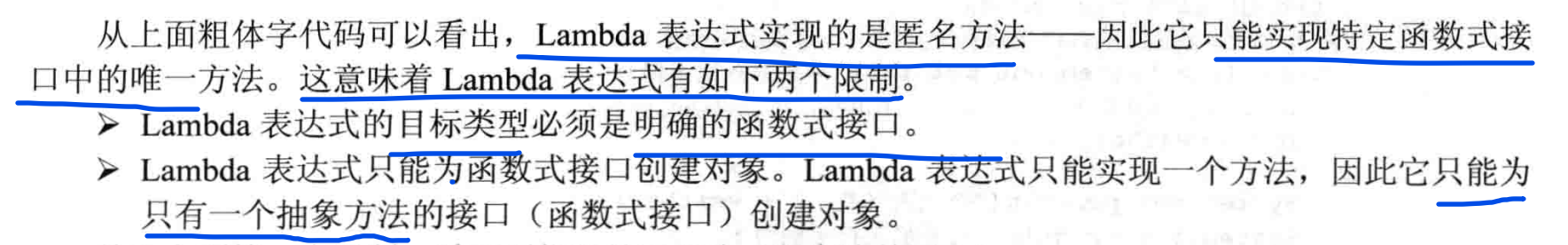
System.out.println(“111”);

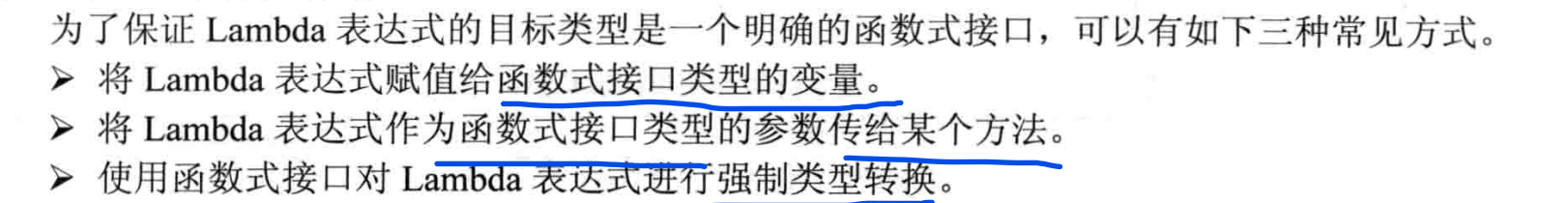
}

2个参数

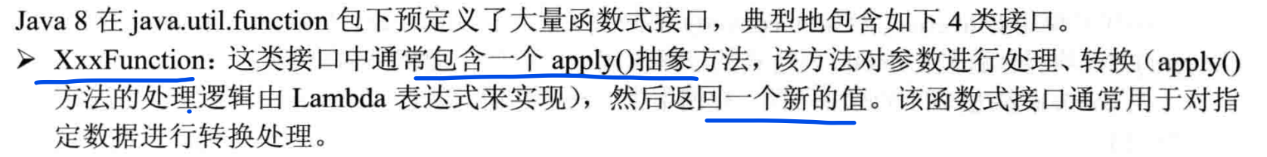
(a,b)->a+b



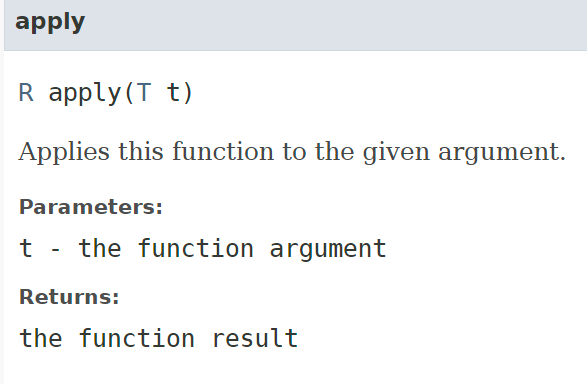


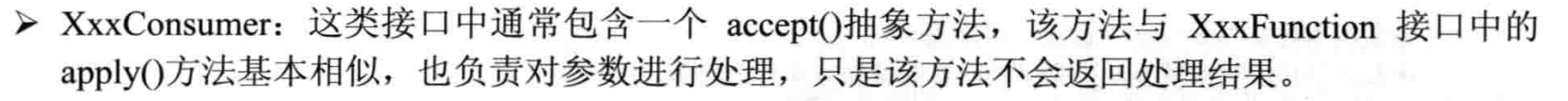


### 函数式接口类型

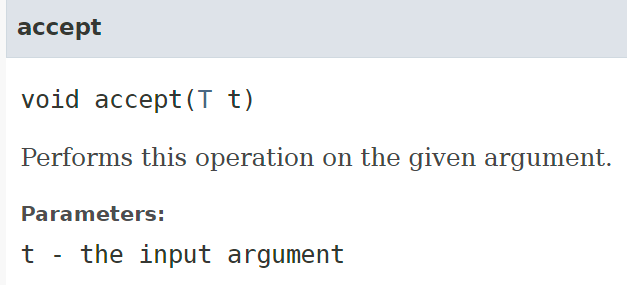


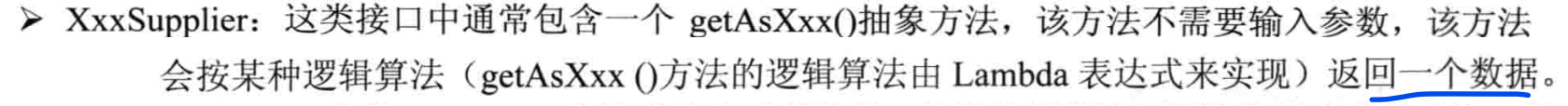
Function函数的表现形式为接收一个参数，并返回一个值。将此函数应用于给定的参数。



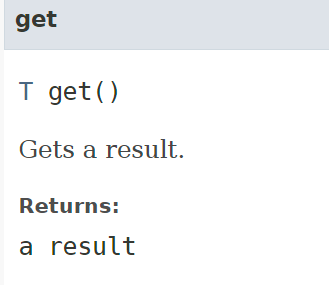


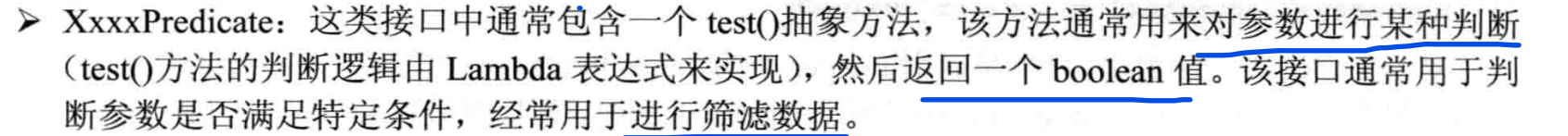
Consumer函数接收一个参数，没有返回值. 对给定参数执行此操作。

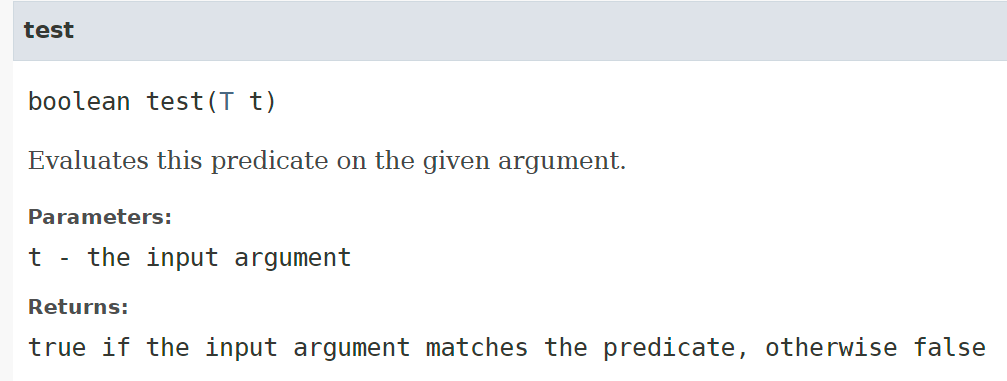




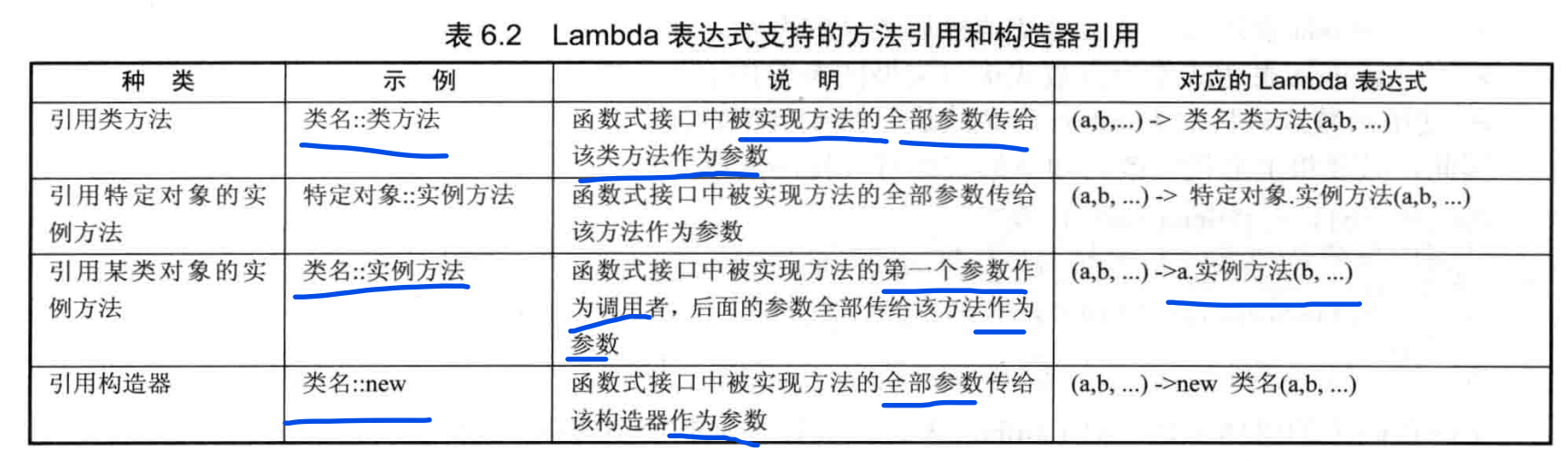
Supplier的表现形式为不接受参数、只返回数据

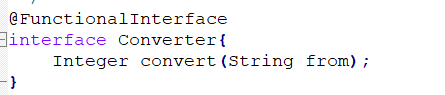






### 方法引用与new对象





#### 类方法引用

Converter converter1 = from -> Integer.valueOf(from);

Converter converter1 = Integer::valueOf;

注意：valueOf是一个static方法

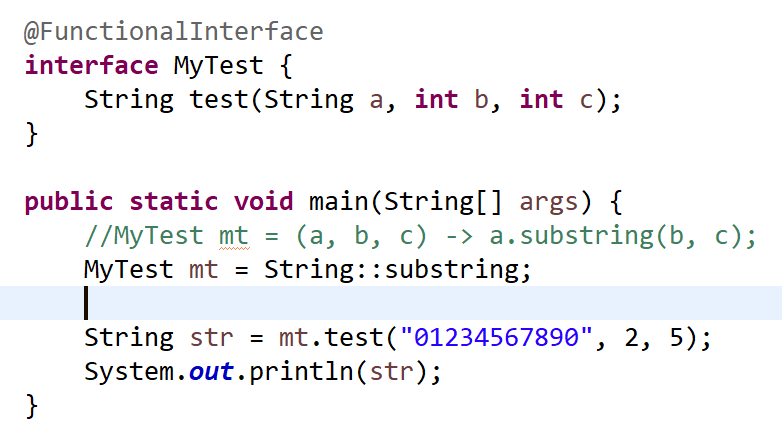
#### 对象引用-1

Converter converter2 = from -> "fkit.org".indexOf(from);

Converter converter2 = new String("fkit.org")::indexOf;

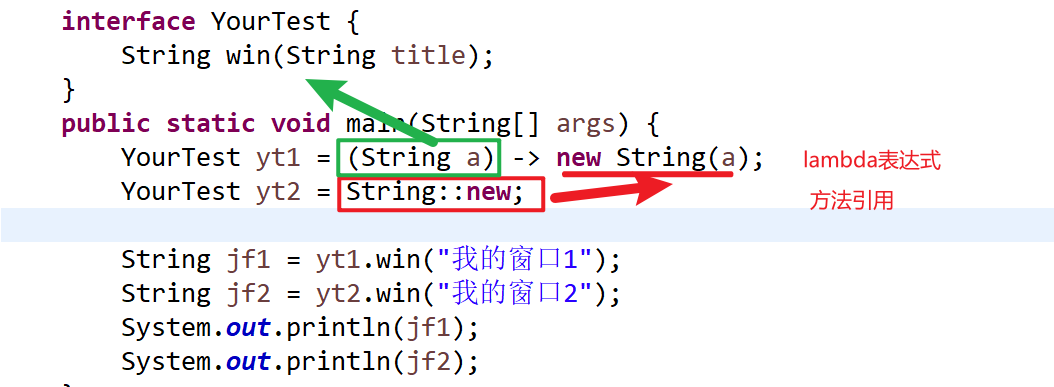
注意：indexOf是一个对象的方法

#### 对象引用-2



注意：MyTest mt = String::substring , substring不是静态方法。表示MyTest函数式接口中第一个参数是指定的对象，其他参数是指定对象的参数。

#### 创建对象



#### 总结

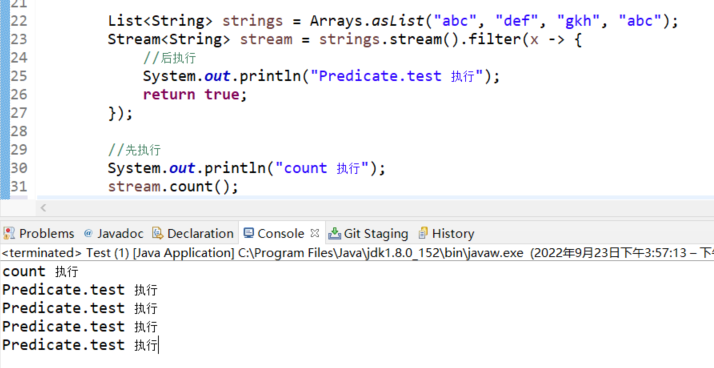
Lambda表达式是函数式接口的实现，是匿名方法的实现。

方法引用只是lambda表达式的实现部分，相比lambda更加简写。

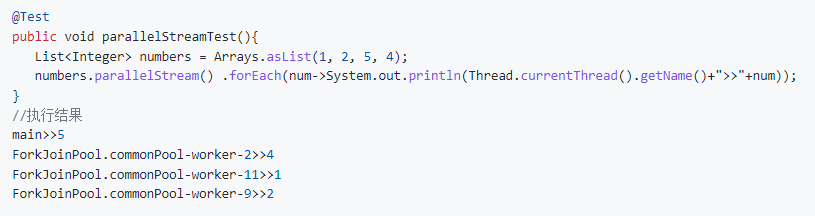
方法引用 其实只是 函数式编程接口的具体实现方法，只是不包含方法名参数等内容。

Lambda与方法引用的作用时一样的，都是函数接口的实现。

在执行返回 Stream 的方法时，并不立刻执行，而是等返回一个非 Stream 的方法后才执行.



多线程执行



## 参考文章

1.[Java8新特性](https://github.com/Snailclimb/JavaGuide/blob/main/docs/java/new-features/java8-common-new-features.md)