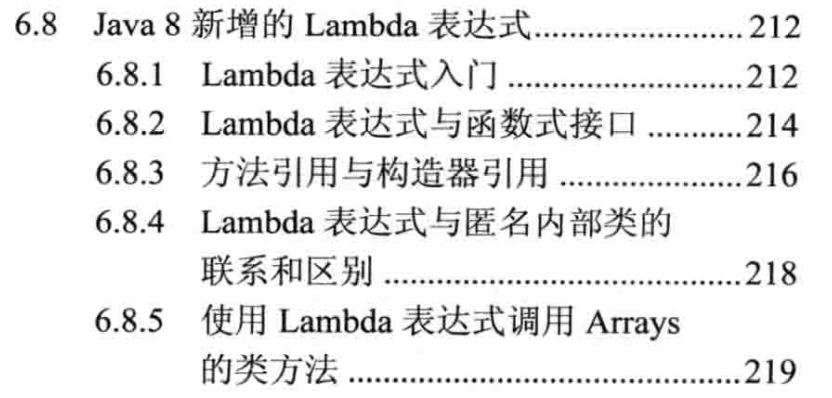
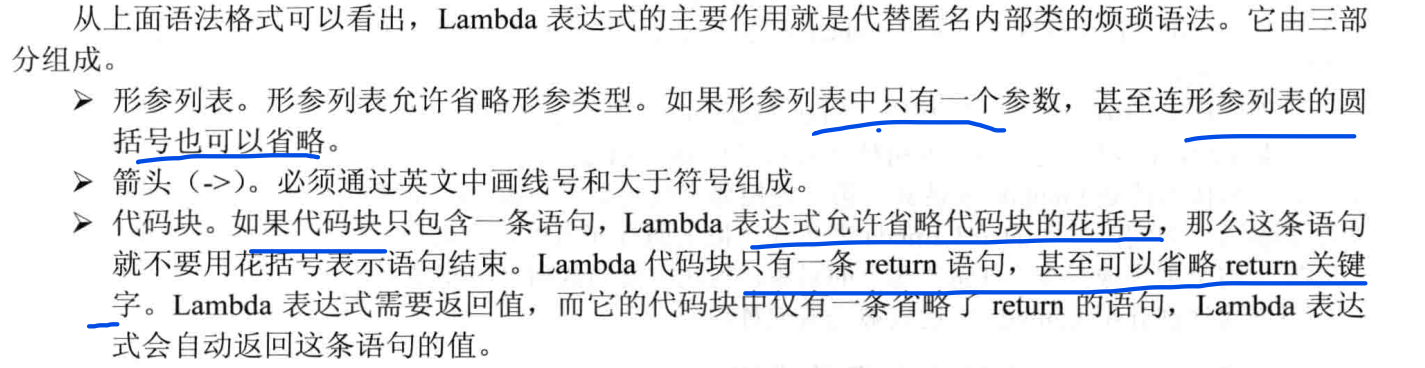


## Lambda表达式



### 语法



无个参数

() -> System.out.println(“111”)

1个参数

Weater ->{

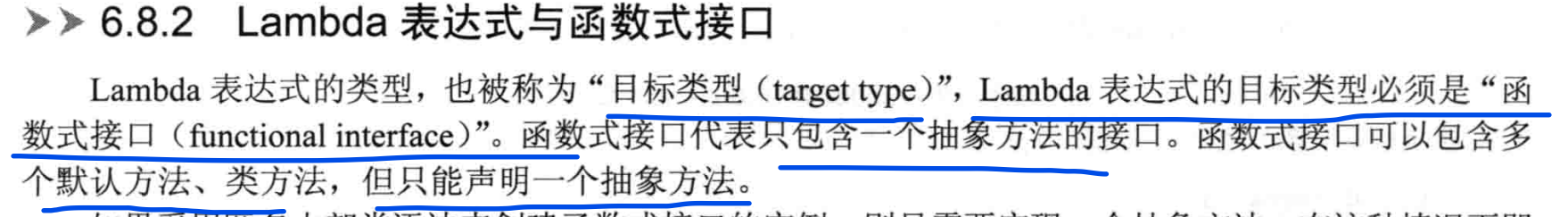
System.out.println(“111”);

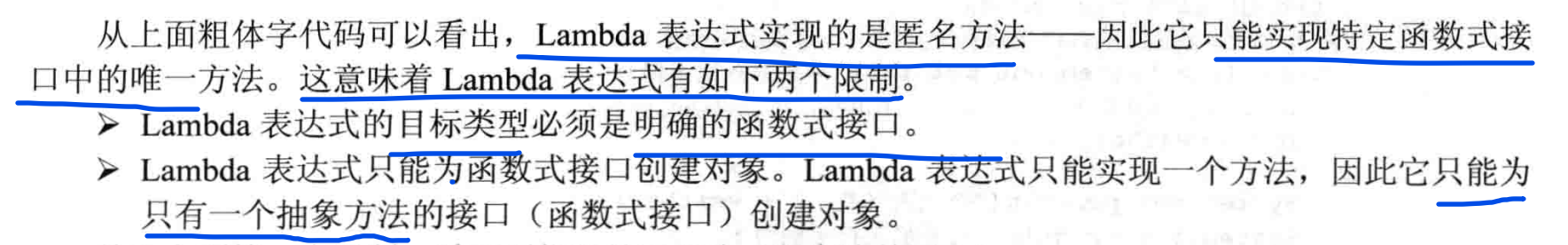
System.out.println(“111”);

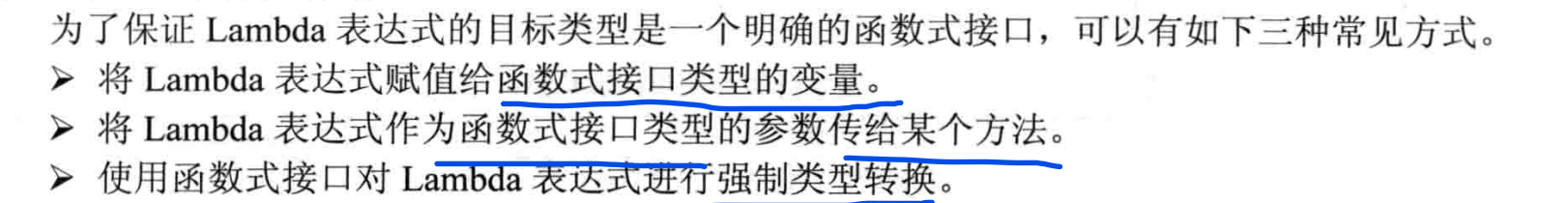
}

2个参数

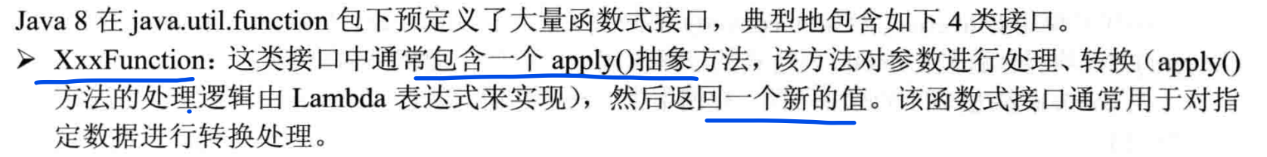
(a,b)->a+b



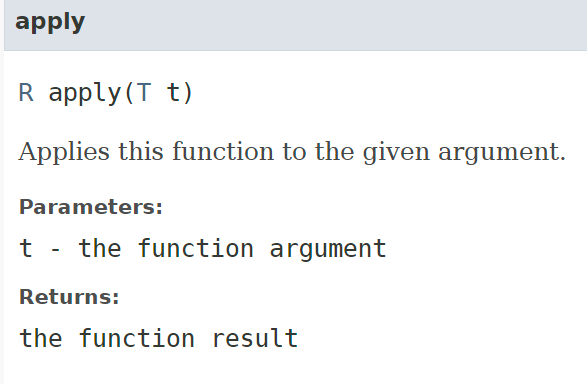


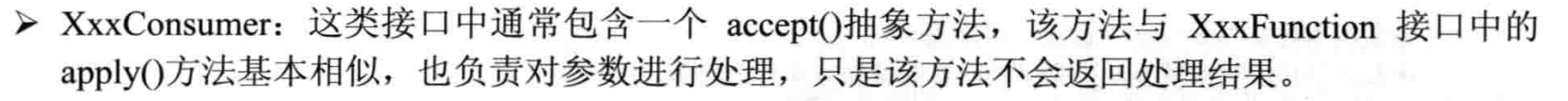


### 函数式接口类型

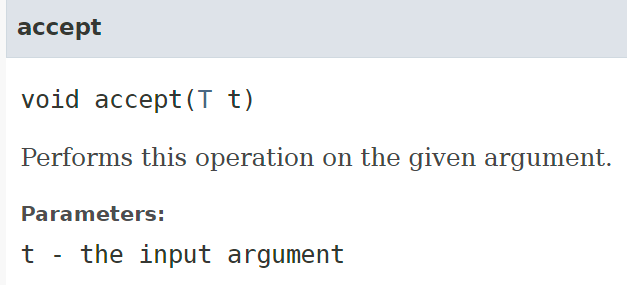


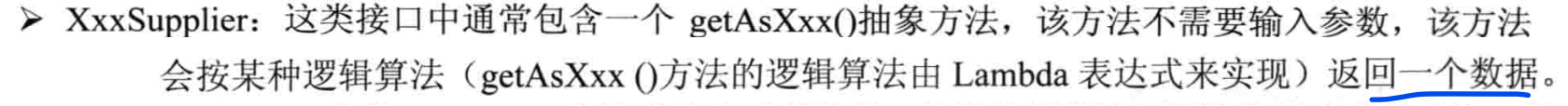
Function函数的表现形式为接收一个参数，并返回一个值。将此函数应用于给定的参数。



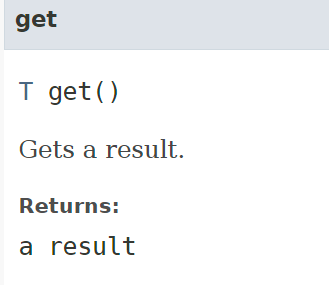


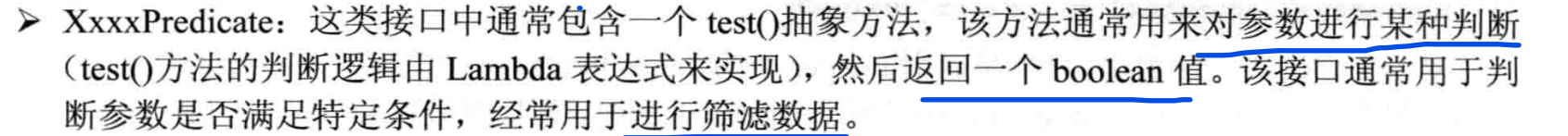
Consumer函数接收一个参数，没有返回值. 对给定参数执行此操作。

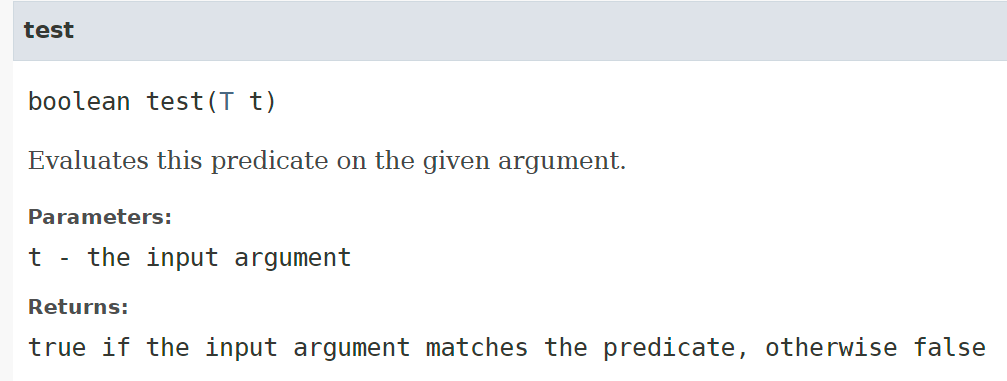




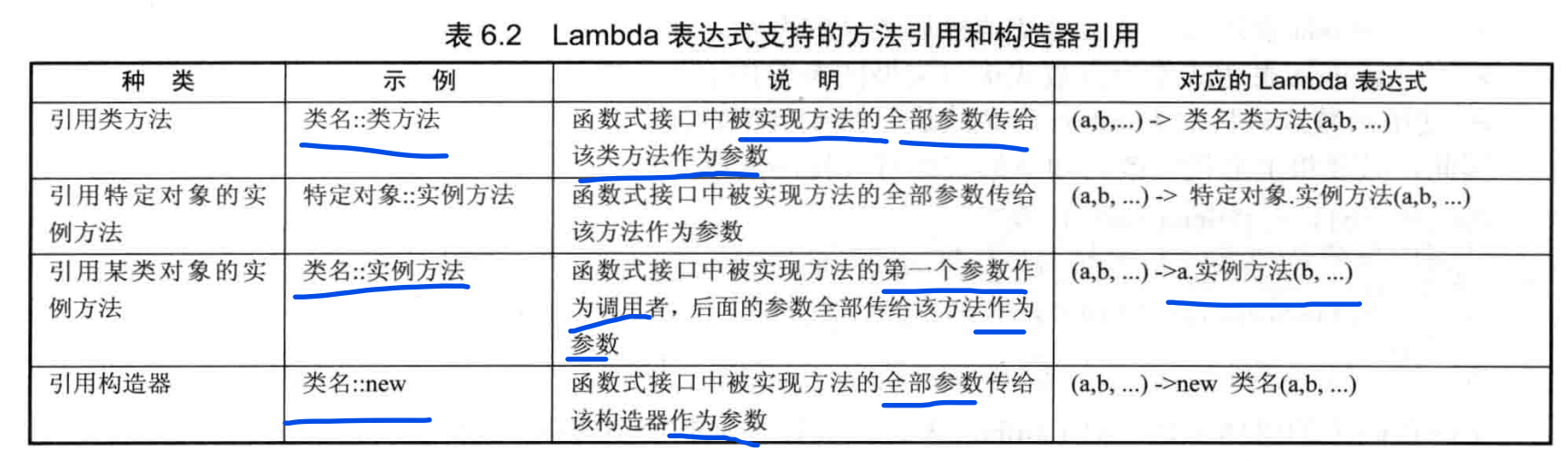
Supplier的表现形式为不接受参数、只返回数据

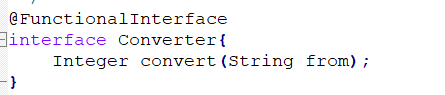






### 方法引用与new对象





#### 类方法引用

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

Converter converter1 = Integer::valueOf;

注意：valueOf是一个static方法

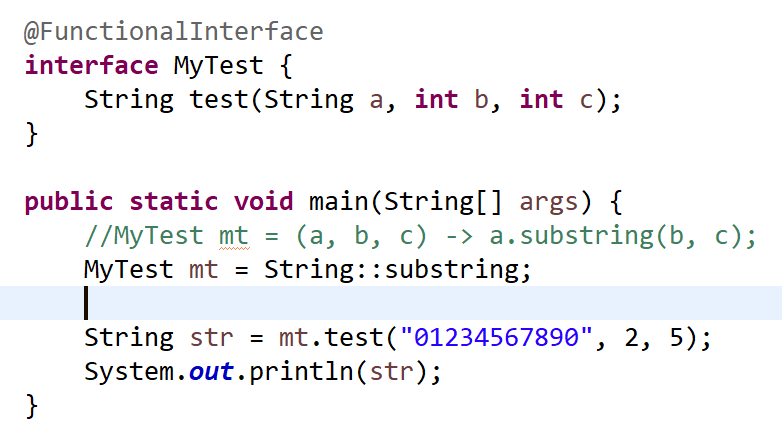
#### 对象引用-1

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

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

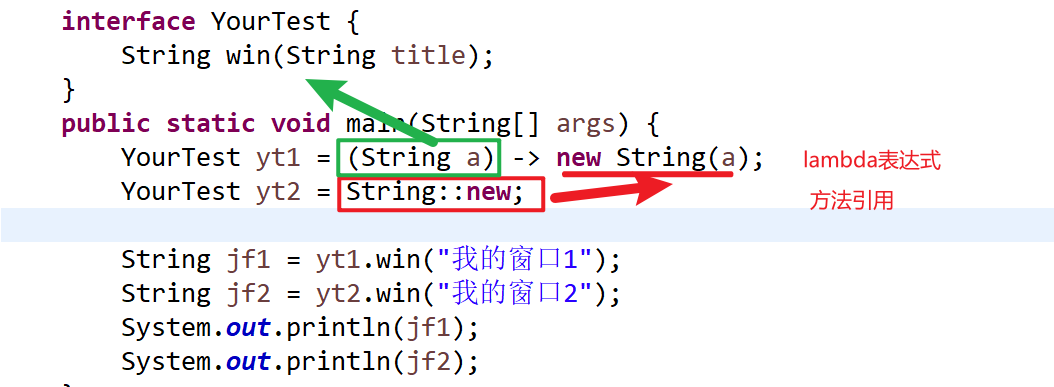
注意：indexOf是一个对象的方法

#### 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

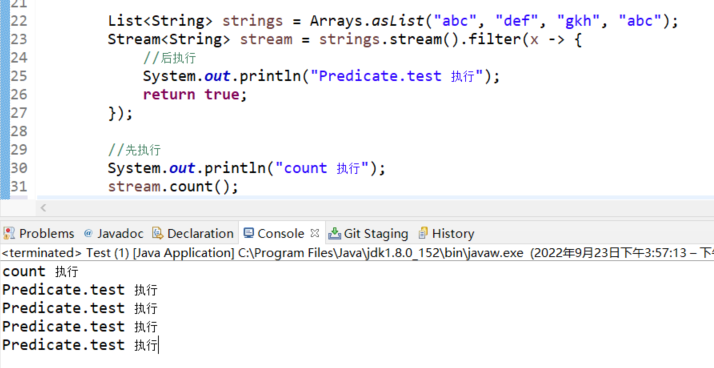
Lambda表达式是函数式接口的实现，是匿名方法的实现。

方法引用只是lambda表达式的实现部分，相比lambda更加简写。

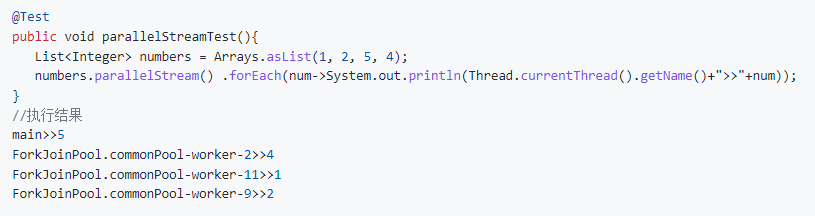
方法引用 其实只是 函数式编程接口的具体实现方法，只是不包含方法名参数等内容。

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)