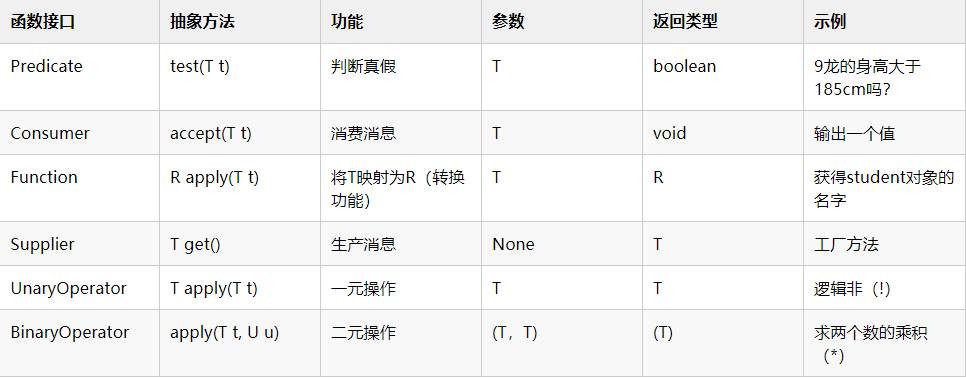
**函数接口是只有一个抽象方法的接口，用作 Lambda 表达式的类型。使用@FunctionalInterface注解修饰的类，编译器会检测该类是否只有一个抽象方法或接口，否则，会报错。可以有多个默认方法，静态方法。**



**惰性求值：只描述Stream，操作的结果也是Stream，这样的操作称为惰性求值。**惰性求值可以像建造者模式一样链式使用，最后再使用及早求值得到最终结果。

**及早求值：得到最终的结果而不是Stream，这样的操作称为及早求值。**

常用的流

collect(Collectors.toList()) 将流转为list。还有toSet()，toMap()等。及早求值。

filter：过滤筛选的作用。内部就是Predicate接口。惰性求值

map：转换功能，内部就是Function接口。惰性求值

flatMap：将多个Stream合并为一个Stream。惰性求值

max和min：在集合中求最大或最小值。及早求值。

count：统计功能，一般都是结合filter使用，因为先筛选出我们需要的再统计即可。及早求值

reduce：实现从一组值中生成一个值。在上述例子中用到的 count 、 min 和 max 方法，因为常用而被纳入标准库中。事实上，这些方法都是 reduce 操作。及早求值。

高级集合类及收集器

转换成值

收集器，一种通用的、从流生成复杂值的结构。只要将它传给 collect 方法，所有的流就都可以使用它了。

转换成块

常用的流操作是将其分解成两个集合，Collectors.partitioningBy帮我们实现了，接收一个Predicate函数式接口。

数据分组

数据分组是一种更自然的分割数据操作，与将数据分成 ture 和 false 两部分不同，可以使用任意值对数据分组。Collectors.groupingBy接收一个Function做转换。

字符串拼接

如果将所有学生的名字拼接起来，怎么做呢？通常只能创建一个StringBuilder，循环拼接。使用Stream，使用Collectors.joining()简单容易。joining接收三个参数，第一个是分界符，第二个是前缀符，第三个是结束符。也可以不传入参数Collectors.joining()，这样就是直接拼接。