链滴

感受 Lambda 之美，推荐收藏，需要时查阅

作者：zxniuniu
原文链接：https://ld246.com/article/1560086704765
来源网站：链滴

一、引言

Java8 最大的特性就是引入 Lambda 表达式，即函数式编程，可以将行为进行传递。**总结就是：使用可变值与函数，函数对不可变值进行处理，映射成另一个值。**

二、函数式接口

1、什么是函数式接口

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 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。

1.1 java8 自带的常用函数式接口。

函数接口类型	抽象方法	功能	参数	返	示例
Predicate	test(T t)	判断真假	T	bo	9 龙的身高大于 185cm 吗?
Consumer	accept(T t)	消费消息	T		输出一个值
Function	R apply(T t)	将 T 映射为 R（转换功能）	T		得 student 对象的名字
Supplier	T get()	生产消息	None	T	厂方法
UnaryOperator	T apply(T t)	一元操作	T	T	

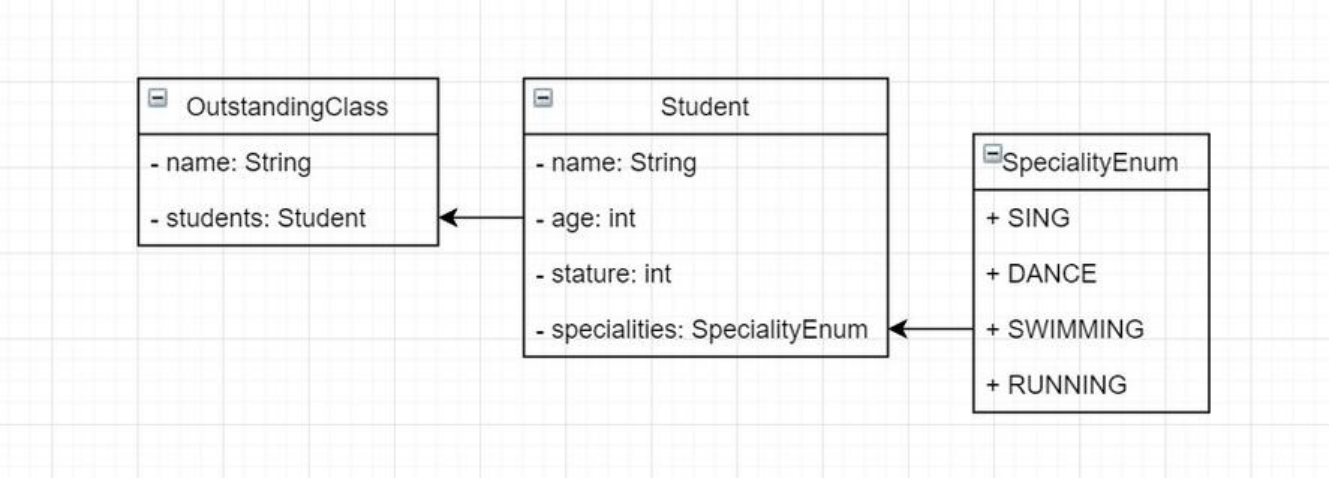
辑非（!）

BinaryOperator apply(T t, U u) 二元操作 (T，T) T) 求两个数的乘积（*）

以上演示了 lambda 接口的使用及自定义一个函数式接口并使用。下面，我们看看 java8 将函数式接封装到流中如何高效的帮助我们处理集合。

注意：Student::getName例子中这种编写 lambda 表达式的方式称为方法引用。格式为ClassNmae::methodName。是不是很神奇，java8 就是这么迷人。

示例：本篇所有示例都基于以下三个类。OutstandingClass：班级；Student：学生；SpecialityEum：特长。



1.2 惰性求值与及早求值

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。

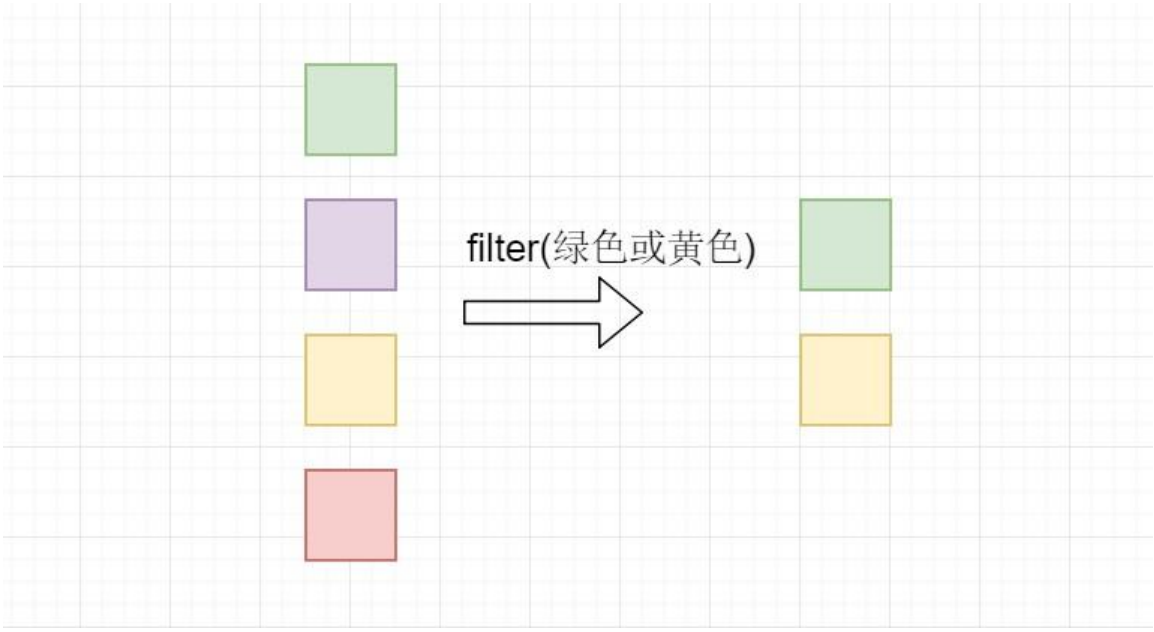
2、常用的流

2.1 collect(Collectors.toList())

将流转换为 list。还有 toSet()，toMap()等。及早求值。

2.2 filter

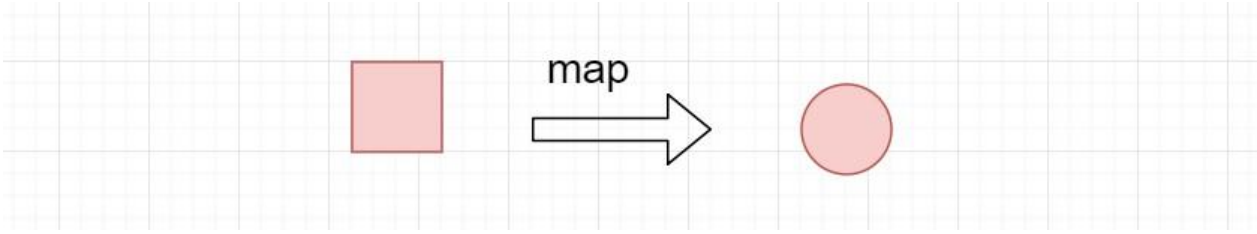
顾名思义，起**过滤筛选**的作用。**内部就是 Predicate 接口。惰性求值。**



比如我们筛选出出身高小于 180 的同学。

2.3 map

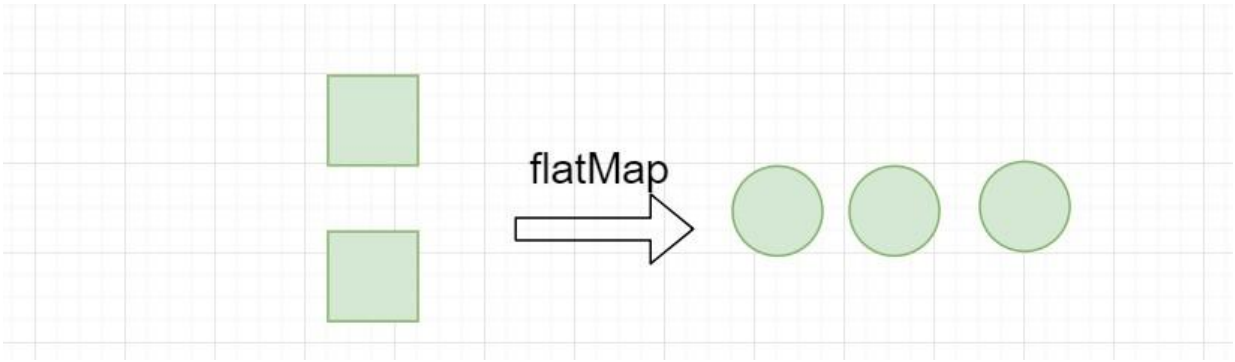
转换功能，内部就是 Function 接口。惰性求值



例子中将 student 对象转换为 String 对象，获取 student 的名字。

2.4 flatMap

将多个 Stream 合并为一个 Stream。惰性求值



调用 Stream.of 的静态方法将两个 list 转换为 Stream，再通过 flatMap 将两个流合并为一个。

2.5 max 和 min

我们经常会在集合中**求最大或最小值**，使用流就很方便。**及早求值。**

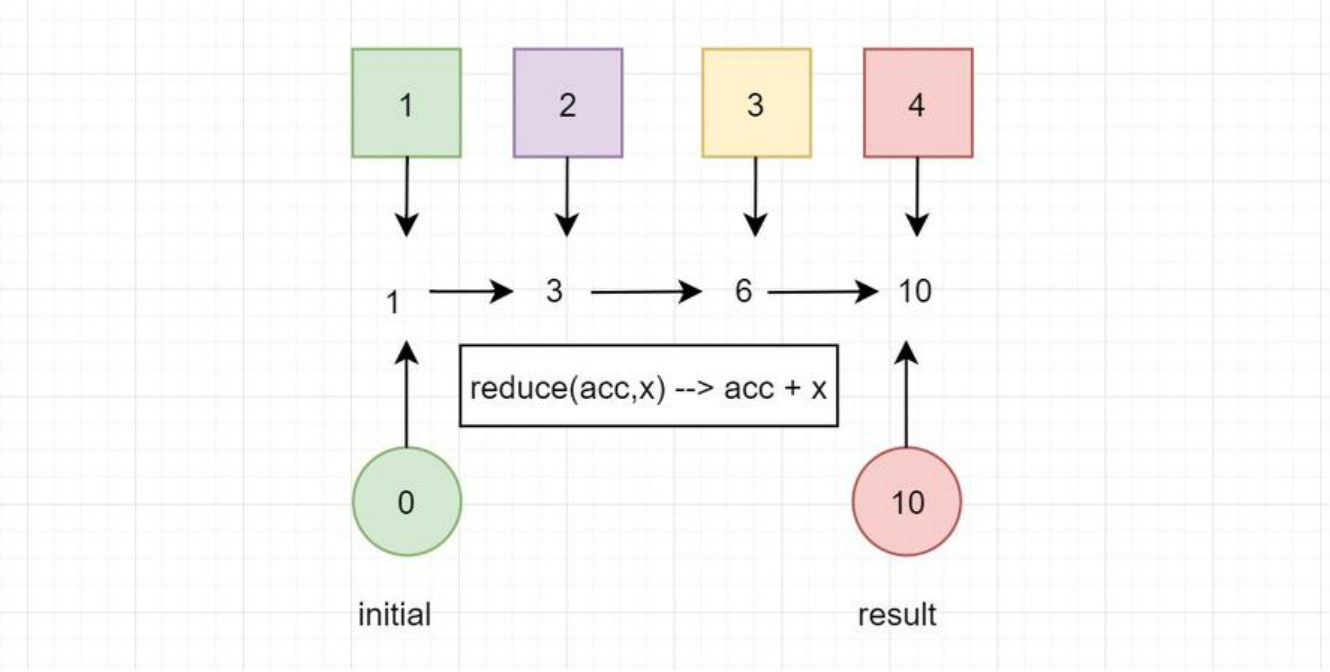
max、min 接收一个 Comparator（例子中使用 java8 自带的静态函数，只需要传进需要比较值即。）并且返回一个 Optional 对象，该对象是 java8 新增的类，专门为了防止 null 引发的空指针异常可以使用 max.isPresent()判断是否有值；可以使用 max.orElse(new Student())，当值为 null 时就用给定值；也可以使用 max.orElseGet(() -> new Student());这需要传入一个 Supplier 的 lambda 达式。

2.6 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

2.7 reduce

reduce 操作可以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。**



我们看得 reduce 接收了一个初始值为 0 的累加器，依次取出值与累加器相加，最后累加器的值就是终的结果。

三、高级集合类及收集器

3.1 转换成值

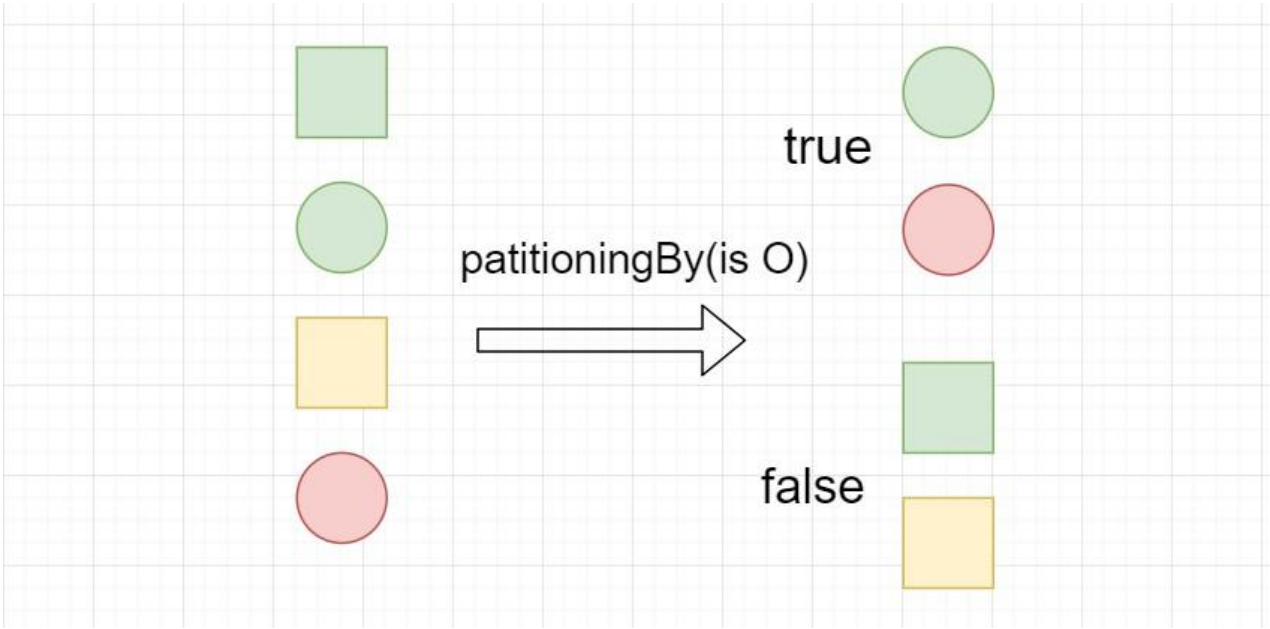
收集器，一种通用的、从流生成复杂值的结构。只要将它传给 collect 方法，所有的流就都可以使用它了。标准类库已经提供了一些有用的收集器，**以下示例代码中的收集器都是从 jav**

.util.stream.Collectors 类中静态导入的。

maxBy 或者 minBy 就是求最大值与最小值。

3.2 转换成块

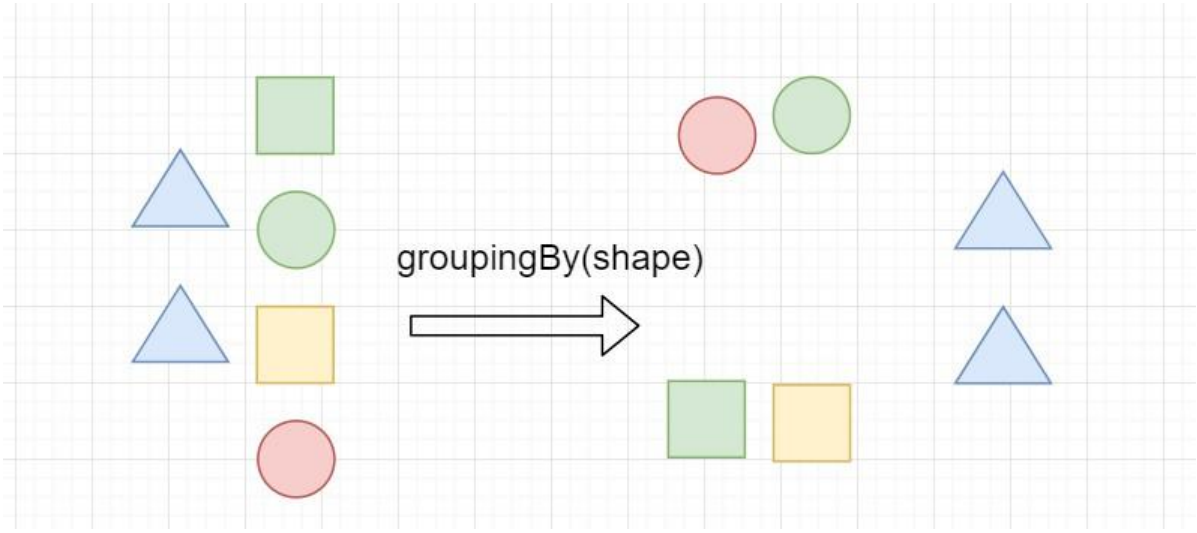
常用的流操作是将其分解成两个集合，Collectors.partitioningBy 帮我们实现了，接收一个 Predicate 函数式接口。



将示例学生分为会唱歌与不会唱歌的两个集合。

3.3 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。**



如图，我们使用 groupingBy 将根据进行分组为圆形一组，三角形一组，正方形一组。

例子：根据学生第一个特长进行分组

分组，并排序

Collectors.groupingBy 与 SQL 中的 group by 操作是一样的。

3.4 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。

四、总结

本篇主要从实际使用讲述了常用的方法及流，使用 java8 可以很清晰表达你要做什么，代码也很简洁本篇例子主要是为了讲解较为简单，大家可以去使用 java8 重构自己现有的代码，自行领会 lambda 奥妙。本文说的 Stream 要组合使用才会发挥更大的功能，链式调用很迷人，根据自己的业务去做吧。

感谢原作者，博客转自：https://juejin.im/post/5ce66801e51d455d850d3a4a