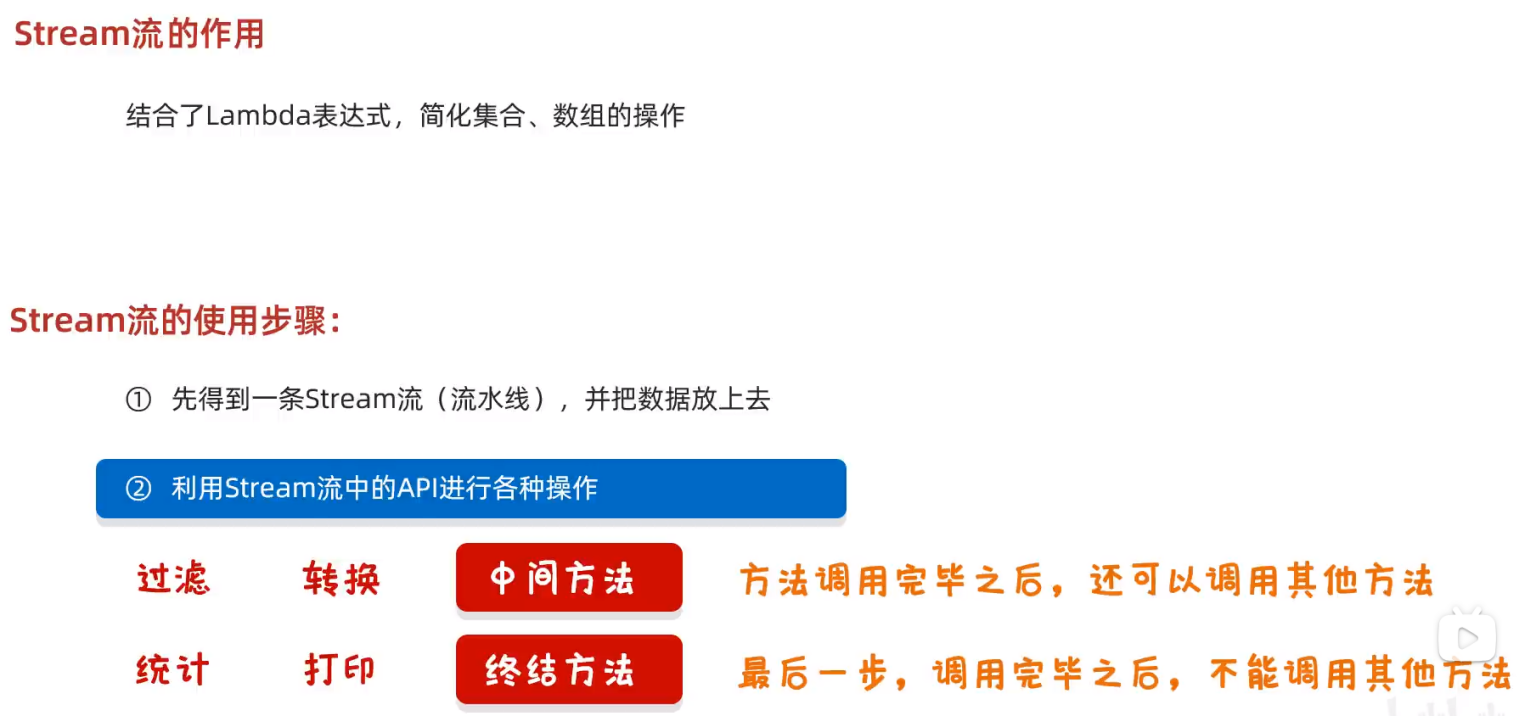
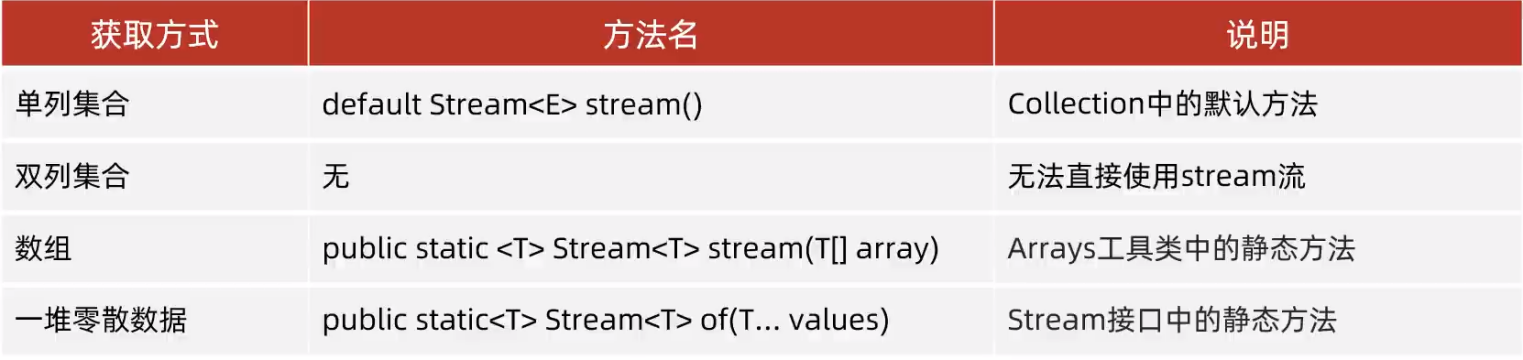
# Stream流



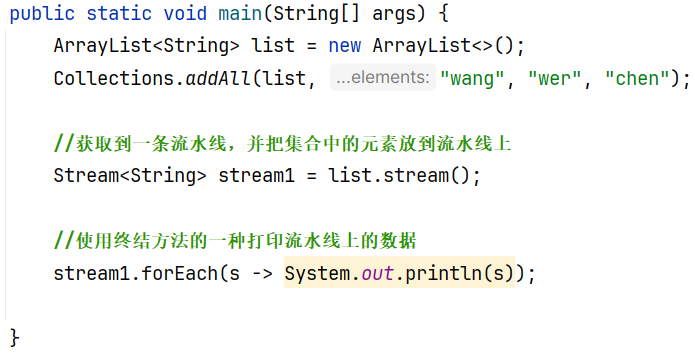
# Stream流的使用步骤

## Stream流放入

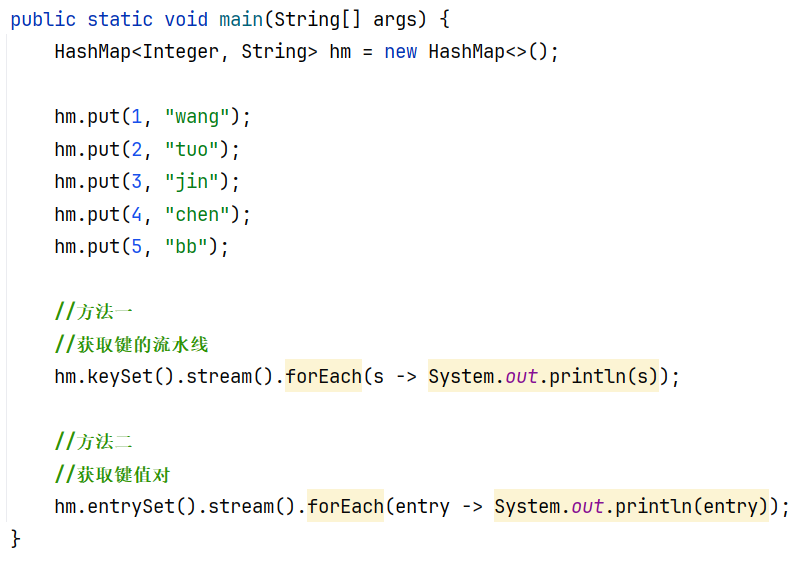
第一步：先获取一条stream流，把数据放入



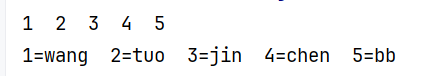
单列集合：



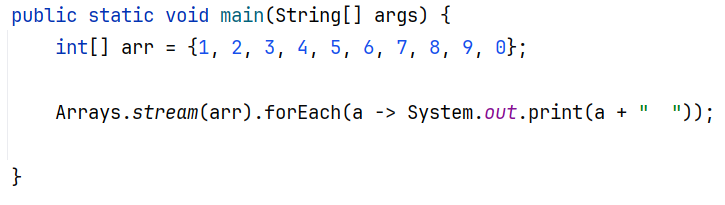
双列集合：不能直接调用stream方法



结果：



数组：需要调用Arrays工具类中的stream方法，把数组作为参数传进去

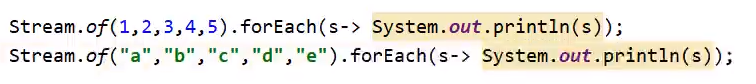


结果：



对于零散数据，如果想使用Stream流，必须满足是同种数据类型

调用Stream接口中的of方法

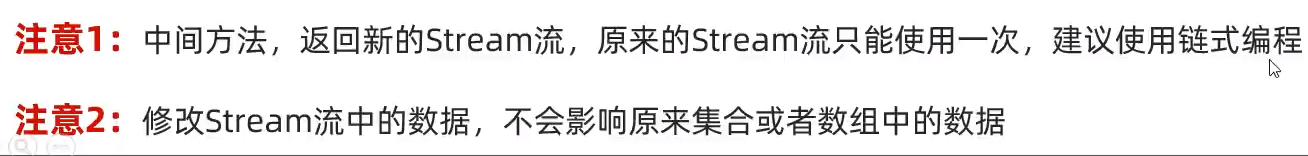


**注意：**

这个方法的参数也可以是数组，但是数组必须是引用数据类型的，如果传递的是基本数据类型的数组，会把整个数组看成一个元素进行操作。

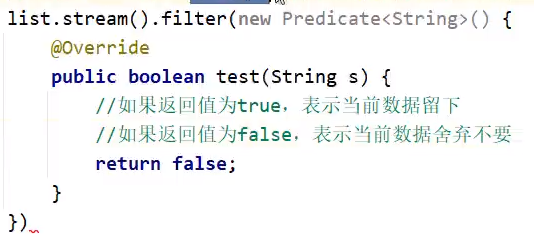
## Stream流的中间方法



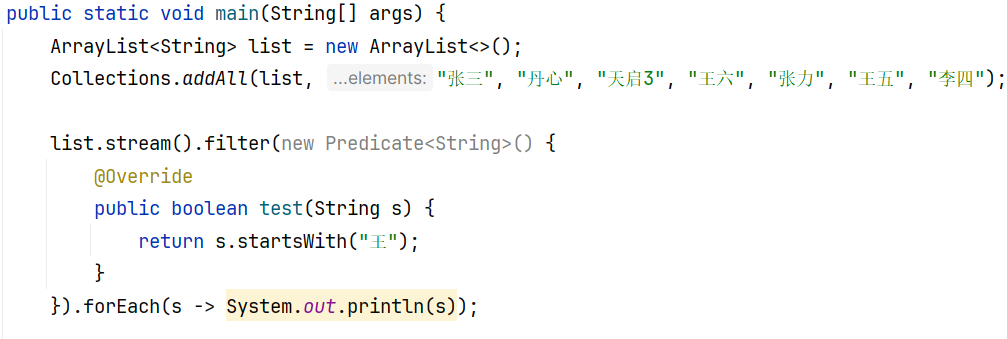


注意1意味着一个Stream流用完一次就关闭了，不能重复使用

## Filter方法



例子：



Filter中使用的是函数式接口Predicate

可以用lambda表达式简化，如下：



## Map方法

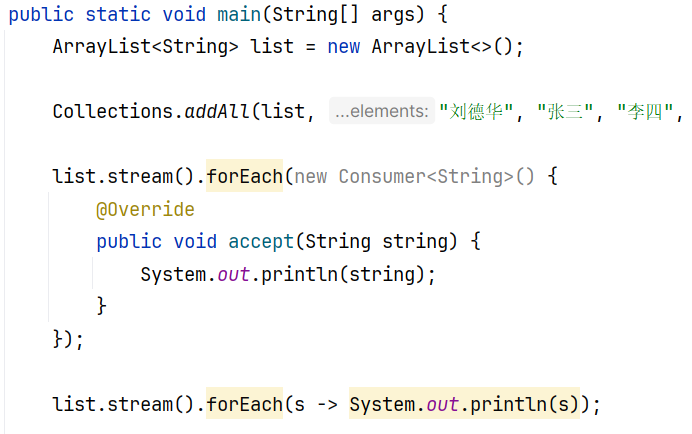
用于类型转换



## Stream流的终结方法

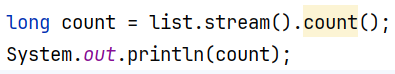


## ForEach方法以及它的lambda表达式

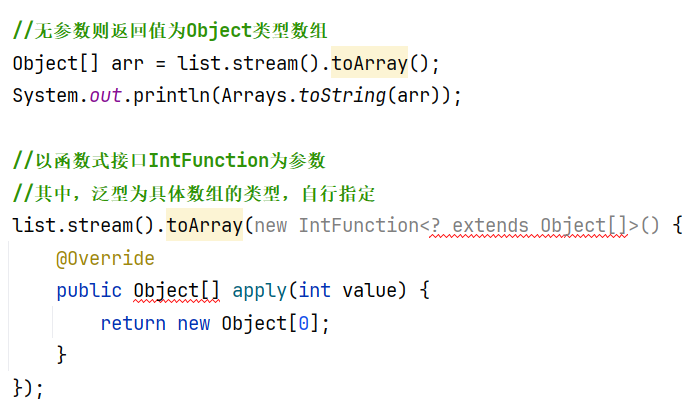


Count方法

用于统计集合中元素的数量，返回值是long



## ToArray方法



指定泛型的一个例子：

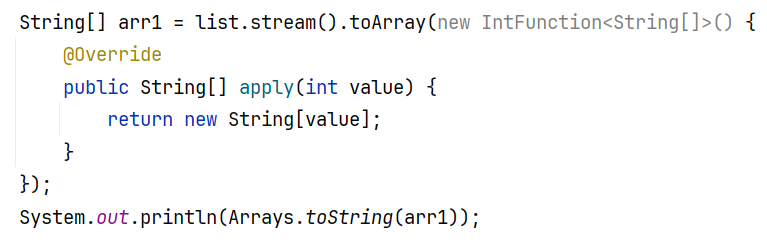
apply的形参:流中数据的个数，要跟数组的长度保持一致

apply的返回值:具体类型的数组

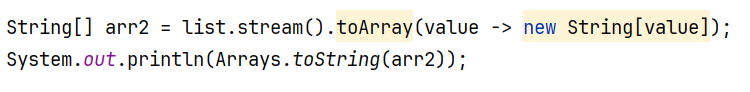
方法体:就是创建数组

toArray方法的参数的作用:负责创建一个指定类型的数组

toArray方法的底层，会依次得到流里面的每一个数据，并把数据放到数组当中toArray方法的返回值:是一个装着流里面所有数据的数组



显然，他也可以用lambda表达式简化，其中value是流水线中元素的个数，也是新数组的长度

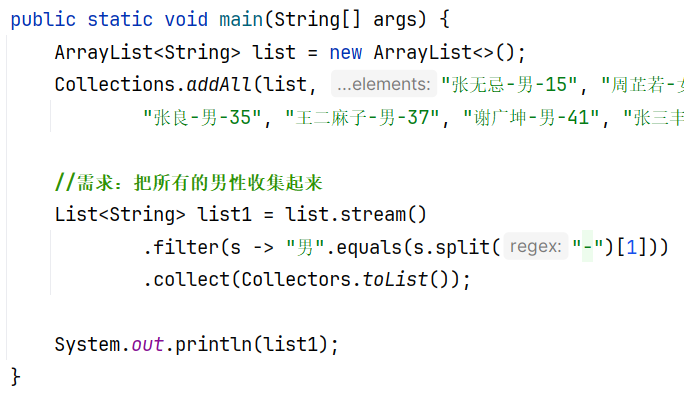


以上结果都是：



## Collect方法

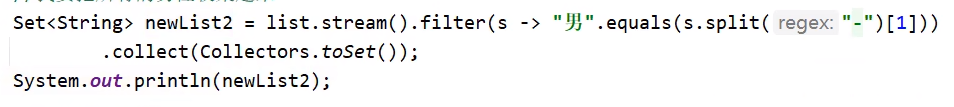
目的在于收集通过中间方法处理过后的数据，放入一个新的集合



结果：



接受者可以是list，set，map



通常以set为接受者，这是为了去除重复元素

**注意，如果要收集到map中，需要指明谁作为键，谁作为值**

toMap 具体细节：

参数一表示键的生成规则

参数二表示值的生成规则

参数一:

Function

泛型一:表示流中每一个数据的类型

泛型二:表示Map集合中键的数据类型

方法apply形参:依次表示流里面的每一个数据

方法体:生成键的代码

返回值:已经生成的键

参数二:

Function

泛型一:表示流中每一个数据的类型

泛型二:表示Map集合中值的数据类型

方法apply形参:依次表示流里面的每一个数据

方法体:生成值的代码

返回值:已经生成的值

此外，作为键的值不能有重复的

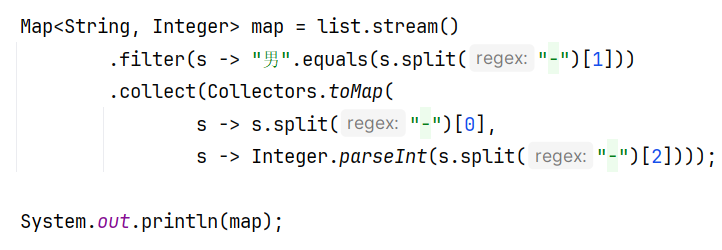
例子：



结果：



同样，collect也可以用lambda表达式简化



# 总结

