Stream流&方法引用

### 一、学习目标

1. 能够理解流与集合相比的优点
2. 能够理解流的延迟执行特点
3. 能够通过集合、映射或数组获取流
4. 能够掌握常用的流操作
5. 能够使用输出语句的方法引用
6. 能够通过4种方式使用方法引用
7. 能够使用类和数组的构造器引用

### 二、学习指南

1. Stream流与集合的对比
2. 方法引用简化Lambda表达式

### 第一堂课

本节知识点

01\_今日内容介绍

02\_使用传统的方式,遍历集合,对集合中的数据进行过滤

03\_使用Stream流的方式,遍历集合,对集合中的数据进行过滤

04\_流式思想概述

视频时间

22分54秒

本节目标

了解今天所学内容

理解传统循环遍历的弊端

熟悉Stream流代码的基本书写

理解Stream流式思想

#### 今日内容介绍

##### 1.1.1知识概述

* 今日内容介绍

##### 1.1.2视频详情



##### 1.1.3总结与补充

无

##### 1.1.4课堂提问与练习

无

##### 1.1.5习题答案

无

#### 使用传统的方式,遍历集合,对集合中的数据进行过滤

##### 1.2.1知识概述

* 通过传统集合对数据操作的复杂, 引入Stream流

##### 1.2.2视频详情



##### 1.2.3总结与补充



这段代码中含有三个循环，每一个作用不同：  
1. 首先筛选所有姓张的人；  
2. 然后筛选名字有三个字的人；  
3. 最后进行对结果进行打印输出。  
每当我们需要对集合中的元素进行操作的时候，总是需要进行循环、循环、再循环。这是理所当然的么？不是。循  
环是做事情的方式，而不是目的。另一方面，使用线性循环就意味着只能遍历一次。如果希望再次遍历，只能再使  
用另一个循环从头开始。

##### 1.2.4课堂提问与练习

无

##### 1.2.5习题答案

无

#### 使用Stream流的方式,遍历集合,对集合中的数据进行过滤

##### 1.3.1知识概述

* 使用Stream流对传统集合进行优化

##### 1.3.2视频详情



##### 1.3.3总结与补充



##### 1.3.4课堂提问与练习

无

##### 1.3.5习题答案

无

#### 流式思想概述

##### 1.4.1知识概述

* 流式思想类似于工厂车间的“生产流水线

##### 1.4.2视频详情



##### 1.4.3总结与补充



元素是特定类型的对象，形成一个队列。 Java中的Stream并不会存储元素，而是按需计算。  
 数据源 流的来源。 可以是集合，数组 等

当使用一个流的时候，通常包括三个基本步骤：获取一个数据源（source）→ 数据转换→执行操作获取想要的结  
 果，每次转换原有 Stream 对象不改变，返回一个新的 Stream 对象（可以有多次转换），这就允许对其操作可以  
 像链条一样排列，变成一个管道。

##### 1.4.4课堂提问与练习

无

##### 1.4.5习题答案

无

### 第二堂课

本节知识点：

05\_两种获取Stream流的方式

06\_Stream流中的常用方法\_forEach

07\_Stream流中的常用方法\_filter

08\_Stream流的特点\_只能使用一次

视频时间：

24分10秒

本节目标 ：

1. 掌握Stream流的两种获取方式
2. 掌握forEach方法的作用
3. 掌握filter方法的作用
4. 理解Stream流只能使用一次的特点

#### 两种获取Stream流的方式

##### 2.1.1知识概述

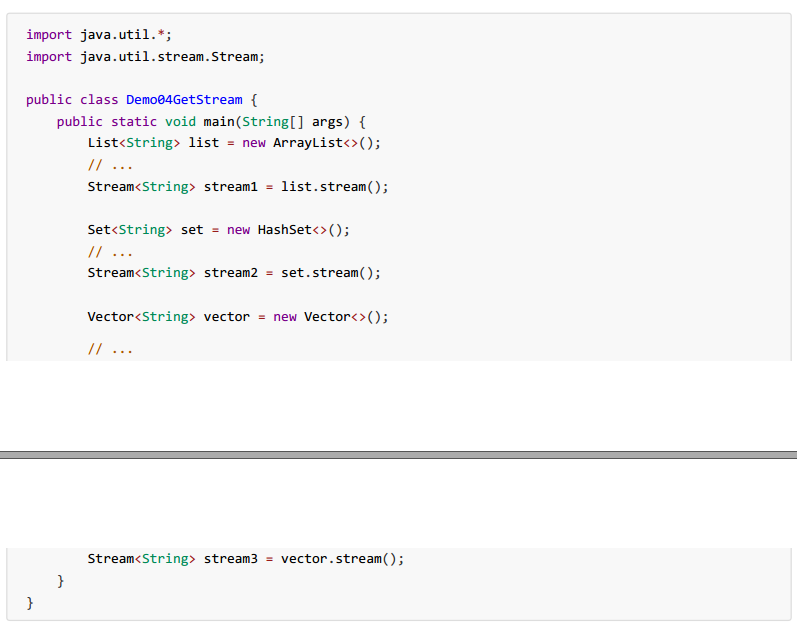
* Stream流的两种获取方式

##### 2.1.2视频详情



##### 2.1.3总结与补充

所有的 Collection 集合都可以通过 stream 默认方法获取流；  
Stream 接口的静态方法 of 可以获取数组对应的流。



##### 2.1.5习题答案

无

#### Stream流中的常用方法\_forEach

##### 2.2.1知识概述

* forEach方法的基本使用

##### 2.2.2视频详情



##### 2.2.3总结与补充



##### 2.2.4课堂提问与练习

无

##### 2.2.5习题答案

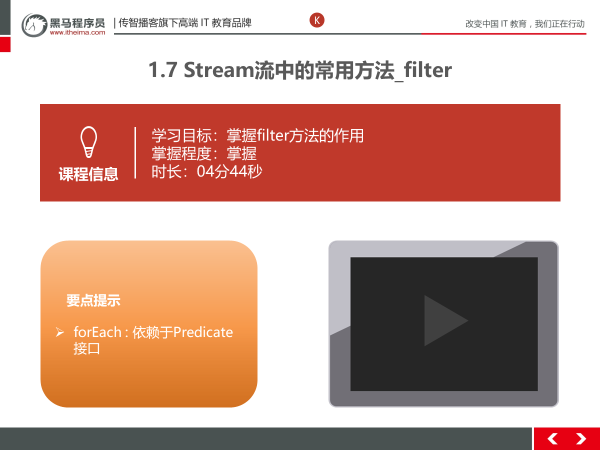
无

#### Stream流中的常用方法\_filter

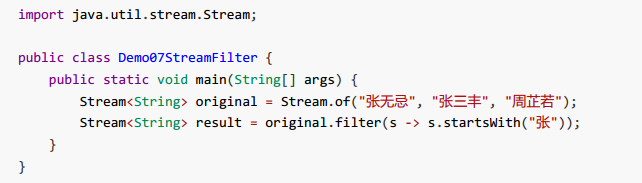
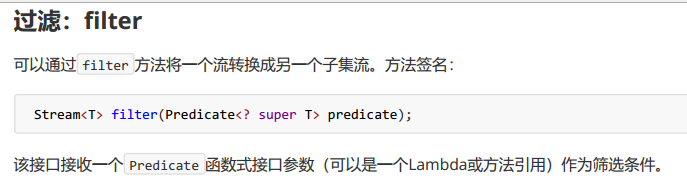
##### 2.3.1知识概述

* filter方法的基本使用

##### 2.3.2视频详情



##### 2.3.3总结与补充



##### 2.3.4课堂提问与练习

无

##### 2.3.5习题答案

无

#### Stream流的特点\_只能使用一次

##### 2.4.1知识概述

* stream流的注意事项, 只能使用一次

##### 2.4.2视频详情



##### 2.4.3总结与补充

Stream流属于管道流,只能被消费(使用)一次  
第一个Stream流调用完毕方法,数据就会流转到下一个Stream上  
而这时第一个Stream流已经使用完毕,就会关闭了  
所以第一个Stream流就不能再调用方法了  
IllegalStateException: stream has already been operated upon or closed

##### 2.4.4课堂提问与练习

无

##### 2.4.5习题答案

无

### 第三堂课

本节知识点：

09\_Stream流中的常用方法\_map

10\_Stream流中的常用方法\_count

11\_Stream流中的常用方法\_limit

12\_Stream流中的常用方法\_skip

13\_Stream流中的常用方法\_concat

视频时间:

16分10秒

本节目标 ：

1. 掌握map方法的使用
2. 掌握终结性方法count的使用
3. 掌握limit方法的使用
4. 掌握skip方法的使用
5. 掌握concat方法的使用

#### Stream流中的常用方法\_map

##### 3.1.1知识概述

* map : 将一个流中的数据映射到另一个流中

##### 3.1.2视频详情



##### 3.1.3总结与补充



##### 3.1.4课堂提问与练习

无

##### 3.1.5习题答案

无

#### Stream流中的常用方法\_count

##### 3.2.1知识概述

* count : 统计流中元素的个数

##### 3.2.2视频详情



##### 3.2.3总结与补充



##### 3.2.4课堂提问与练习

无

##### 3.2.5习题答案

无

#### Stream流中的常用方法\_limit

##### 3.3.1知识概述

* limt : 截取

##### 3.3.2视频详情



##### 3.3.3总结与补充

limit 方法可以对流进行截取，只取用前n个。



##### 3.3.4课堂提问与练习

无

##### 3.3.5习题答案

无

#### Stream流中的常用方法\_skip

##### 3.4.1知识概述

* skip : 跳过

##### 3.4.2视频详情



##### 3.4.3总结与补充

如果希望跳过前几个元素，可以使用 skip 方法获取一个截取之后的新流



##### 3.4.4课堂提问与练习

无

##### 3.4.5习题答案

无

#### Stream流中的常用方法\_concat

##### 3.5.1知识概述

* concat : 拼接流对象

##### 3.5.2视频详情



##### 3.5.3总结与补充

有两个流，希望合并成为一个流，那么可以使用 Stream 接口的静态方法 concat



##### 3.5.4课堂提问与练习

无

##### 3.5.5习题答案

无

### 第四堂课

本节知识点：

14\_练习：集合元素处理（传统方式）

15\_练习：集合元素处理（Stream方式）

16\_方法引用基本介绍

视频时间：

21分42秒

本节目标 ：

掌握集合传统方式对数据的操作

掌握Stream流方式对数据的操作

理解方法引用是对Lambda的简化书写

#### 4.1 练习：集合元素处理（传统方式）

##### 4.1.1知识概述

* 利用传统集合的方式, 对数据进行筛选、跳过..

##### 4.1.2视频详情



##### 4.1.3总结与补充

现在有两个 ArrayList 集合存储队伍当中的多个成员姓名，要求使用传统的for循环（或增强for循环）依次进行以  
下若干操作步骤：  
1. 第一个队伍只要名字为3个字的成员姓名；存储到一个新集合中。  
2. 第一个队伍筛选之后只要前3个人；存储到一个新集合中。  
3. 第二个队伍只要姓张的成员姓名；存储到一个新集合中。  
4. 第二个队伍筛选之后不要前2个人；存储到一个新集合中。  
5. 将两个队伍合并为一个队伍；存储到一个新集合中。  
6. 根据姓名创建 Person 对象；存储到一个新集合中。  
7. 打印整个队伍的Person对象信息。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64 | public class Demo01StreamTest {  public static void main(String[] args) {  //第一支队伍  ArrayList<String> one = new ArrayList<>();  one.add("迪丽热巴");  one.add("宋远桥");  one.add("苏星河");  one.add("石破天");  one.add("石中玉");  one.add("老子");  one.add("庄子");  one.add("洪七公");  //1. 第一个队伍只要名字为3个字的成员姓名；存储到一个新集合中。  ArrayList<String> one1 = new ArrayList<>();  for (String name : one) {  if(name.length()==3){  one1.add(name);  }  }  //2. 第一个队伍筛选之后只要前3个人；存储到一个新集合中。  ArrayList<String> one2 = new ArrayList<>();  for (int i = 0; i <3 ; i++) {  one2.add(one1.get(i));//i = 0,1,2  }  //第二支队伍  ArrayList<String> two = new ArrayList<>();  two.add("古力娜扎");  two.add("张无忌");  two.add("赵丽颖");  two.add("张三丰");  two.add("尼古拉斯赵四");  two.add("张天爱");  two.add("张二狗");  //3. 第二个队伍只要姓张的成员姓名；存储到一个新集合中。  ArrayList<String> two1 = new ArrayList<>();  for (String name : two) {  if(name.startsWith("张")){  two1.add(name);  }  }  //4. 第二个队伍筛选之后不要前2个人；存储到一个新集合中。  ArrayList<String> two2 = new ArrayList<>();  for (int i = 2; i <two1.size() ; i++) {  two2.add(two1.get(i)); //i 不包含0 1  }  //5. 将两个队伍合并为一个队伍；存储到一个新集合中。  ArrayList<String> all = new ArrayList<>();  all.addAll(one2);  all.addAll(two2);  //6. 根据姓名创建Person对象；存储到一个新集合中。  ArrayList<Person> list = new ArrayList<>();  for (String name : all) {  list.add(new Person(name));  }  //7. 打印整个队伍的Person对象信息。  for (Person person : list) {  System.out.println(person);  }  }  } |

##### 4.1.4课堂提问与练习

无

##### 4.1.5习题答案

无

#### 4.2 练习：集合元素处理（Stream方式）

##### 4.2.1知识概述

* 利用Stream流优化传统的集合代码

##### 4.2.2视频详情



##### 4.2.3总结与补充



##### 4.2.4课堂提问与练习

无

##### 4.2.5习题答案

无

#### 4.3 方法引用基本介绍

##### 4.3.1知识概述

* 方法引用 : 对Lambda的一种简化写法

##### 4.3.2视频详情



##### 4.3.3总结与补充



##### 4.3.4课堂提问与练习

无

##### 4.3.5习题答案

无

#### 4.4 常用的函数式接口\_Predicate接口\_默认方法or&negate

##### 4.4.1知识概述

* Or : 或者
* negate : 非

##### 4.4.2视频详情



##### 4.4.3总结与补充

与 and 的“与”类似，默认方法 or 实现逻辑关系中的“或”

negate - > 非

##### 4.4.4课堂提问与练习

无

##### 4.4.5习题答案

无

### 第五堂课

本节知识点：

17\_方法引用\_通过对象名引用成员方法

18\_方法引用\_通过类名引用静态成员方法

19\_方法引用\_通过super引用父类的成员方法

视频时间：

24分21秒

本节目标 ：

了解对象名引用成员方法的格式

了解类名引用静态方法的格式

了解super引用父类方法的格式

#### 5.1 方法引用\_通过对象名引用成员方法

##### 5.1.1知识概述

* 对象名:: 方法名

##### 5.1.2视频详情



##### 5.1.3总结与补充

通过对象名引用成员方法  
使用前提是对象名是已经存在的,成员方法也是已经存在  
就可以使用对象名来引用成员方法



##### 5.1.4课堂提问与练习

无

##### 5.1.5习题答案

无

#### 5.2 方法引用\_通过类名引用静态成员方法

##### 5.2.1知识概述

* 类名::方法名

##### 5.2.2视频详情

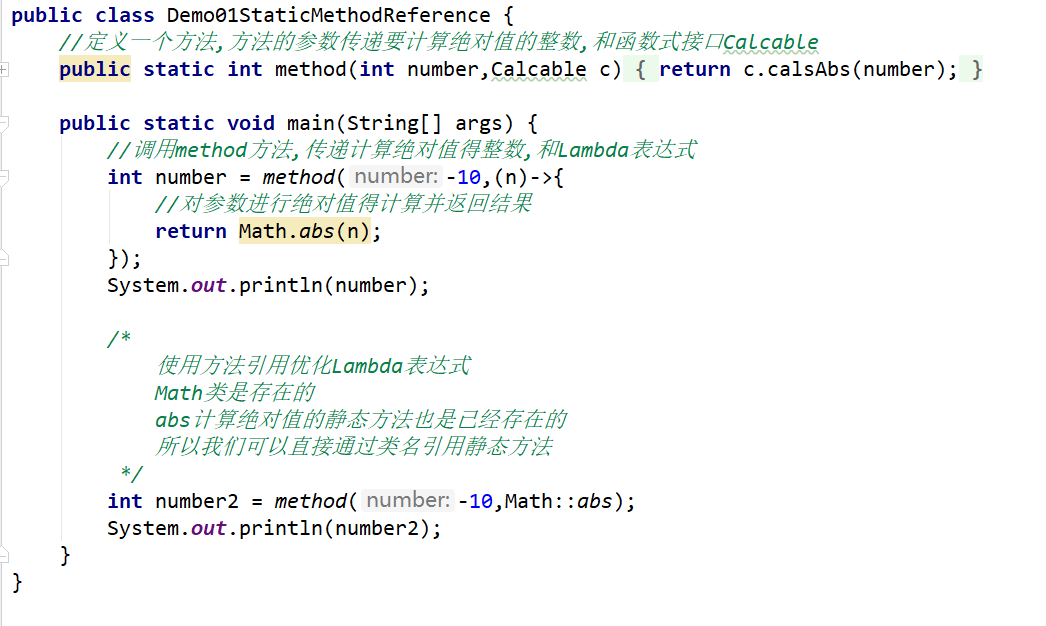


##### 5.2.3总结与补充

通过类名引用静态成员方法

类已经存在,静态成员方法也已经存在

就可以通过类名直接引用静态成员方法



##### 5.2.4课堂提问与练习

无

##### 5.2.5习题答案

无

#### 5.3 通过super引用父类的成员方法

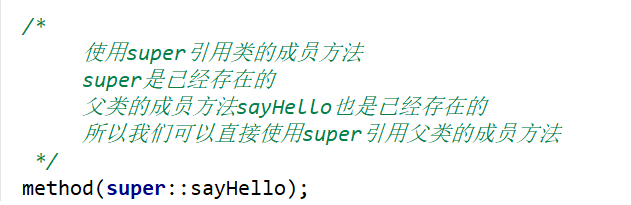
##### 5.3.1知识概述

* super :: 方法名

##### 5.3.2视频详情



##### 5.3.3总结与补充



##### 5.3.4课堂提问与练习

无

##### 5.3.5习题答案

无

### 第六堂课

本节知识点：

20\_方法引用\_通过this引用本类的成员方法

21\_方法引用\_类的构造器(构造方法)引用

22\_方法引用\_数组的构造器引用

视频时间：

22分20秒

本节目标 ：

了解this引用本类方法的格式

了解引用对象构造方法的格式

了解引用数组构造方法的格式

#### 6.1 方法引用\_通过this引用本类的成员方法

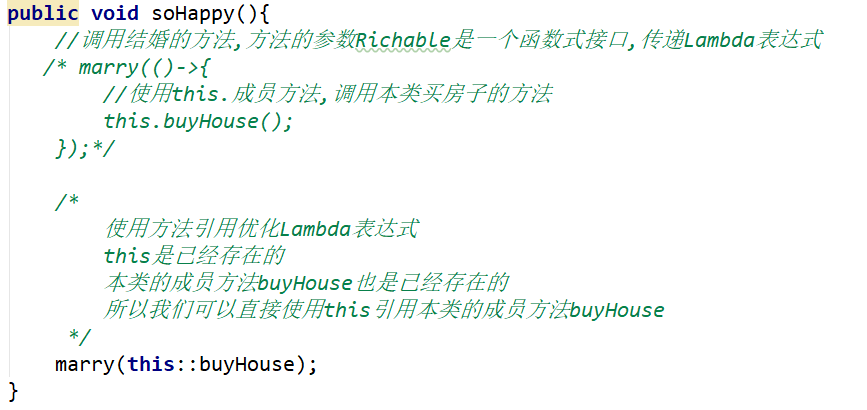
##### 6.1.1知识概述

##### 方法引用\_通过this引用本类的成员方法

##### 5.1.2视频详情



##### 5.1.3总结与补充



##### 6.1.4课堂提问与练习

无

##### 6.1.5习题答案

无

#### 6.2 方法引用\_类的构造器(构造方法)引用

##### 6.2.1知识概述

* 类名::new

##### 6.2.2视频详情



##### 6.2.3总结与补充

##### 6.2.4课堂提问与练习

无

##### 5.2.5习题答案

无

#### 6.3 方法引用\_数组的构造器引用

##### 6.3.1知识概述

* 数组构造器的引用

##### 6.3.2视频详情



##### 6.3.3总结与补充



##### 6.3.4课堂提问与练习

无

##### 6.3.5习题答案

无