流迭代

创建流

一、直接创建

**import** java.util.stream.Stream;

Stream<String> stream = Stream.of("苹果", "哈密瓜", "香蕉", "西瓜", "火龙果");

二、由数组转化

String[] fruitArray = **new** String[] {"苹果", "哈密瓜", "香蕉", "西瓜", "火龙果"};

Stream<String> stream = Stream.of(fruitArray);

（这两种方式本质是一样的，第一种方式其实就是字符串数组）

三、由集合转化

List<String> fruits = **new** ArrayList<>();

fruits.add("苹果");

fruits.add("哈密瓜");

fruits.add("香蕉");

fruits.add("西瓜");

fruits.add("火龙果");

Stream<String> stream = fruits.stream();

迭代流

Stream 提供的迭代方法也叫做 forEach( )

Stream<String> stream = Stream.of("苹果", "哈密瓜", "香蕉", "西瓜", "火龙果");

stream.forEach(System.out::println);

流数据过滤

过滤 期末考试平均分不低于80分 且 无违规记录的学生：

新特性实现：

List<Pupil> pupils = **new** ArrayList<>();

*// 这里假设小学生数据对象已经存入了*

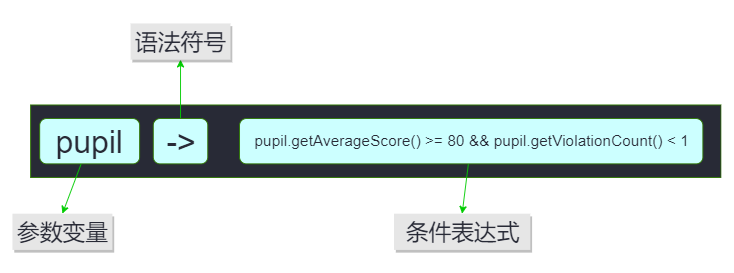
pupils.stream()

.filter(pupil -> pupil.getAverageScore() >= 80 && pupil.getViolationCount() < 1)

.forEach(pupil -> {System.out.println(pupil.getName());});

Filter() 方法

功能：对流中的数据对象进行过滤。



方法参数是一个Lambda表达式，箭头后面是条件语句，判断数据需要符合的条件。

（箭头后的过滤条件语句（非可执行的语句）。传统代码中，条件语句写在（）中，所以新特性条件语句可以用（）而不能用{ }。）

数据流映射

小习题：对于一组数字：

List<Integer> numbers = Arrays.asList(3, 2, 2, 7, 63, 2, 3, 5);

计算每个数字的平方数并输出。

解答

使用Stream API就很简便了：

numbers.stream()

.map(num -> {

**return** num \* num;

})

.forEach(System.out::println);

Map（）方法通常称作**映射**，其作用就是用新的元素替换掉原来**相同位置**的元素。相当于每个对象都经历一次转换。

映射到新数据

Map（）方法的参数是一个Lambda表达式，在语句块中对流中的每个数据对象进行计算、处理，最后用 return 语句返回的对象，就是转换后的对象。

（映射后的对象类型，可以与流中原始的对象类型不一致。）

int sum = students.stream()  
 .map(stu->{return stu.getMidtermScore();})//改变流中的数据类型  
 .reduce((a,b)->a+b)  
 .get();

数据流排序

.sorted 实现排序

students.stream()

*// 实现升序排序*

.sorted((student1, student2) -> {

**return** student1.getRollNo() - student2.getRollNo();

})

.forEach(System.out::println);

Sorted() 顾名思义，就是完成排序的方法。把排序规则写成一个Lambda表达式传给此方法即可。

参数顺序

（student1，student2）中的student1指代后一个元素，student2指代前一个元素。

特殊写法

少数情况下，如果排序计算语句简单、系统能自动识别需要返回的值，代码可以简写为：

.sorted((student1, student2) -> student1.getRollNo() - student2.getRollNo())

初学者还是推荐写完整的

.sorted((... , ...) -> {

... ...

**return** ... ...;

})

数据流摘取

小习题：对于一组数字：

List<Integer> numbers = Arrays.asList(3, 2, 2, 7, 63, 2, 3, 5);

找出最大的前3个数字。

使用排序+摘取接口完成：

numbers.stream()

.sorted((n1, n2) -> n2 - n1)

.limit(3)

.forEach(System.out::println);

Limit（）方法的作用是返回流的 前n个元素，n不能为负。

（不能摘取任意位置哦，只能是流开头的）

Stream.collect()

//1.收集数据到list集合中

stream.collect(Collectors.toList())

//2.收集数据到set集合中

stream.collect(Collectors.toSet())

//3.收集数据到指定的集合中

Collectors.toCollection(Supplier<C> collectionFactory)

stream.collect(Collectors.joining())