(五)java函数式编程技巧reduce概念原理 stream reduce方法详解 reduce三个参数的reduce方法实现使用
1. reduce方法
2. reduce三个参数的reduce方法
3. 三个参数的reduce方法实现

发于 2020-08-27 11:12 阅读(1) 评论(0) 点赞(0) 分享 stream reduce方法详解 stream reduce方法实现 reduce三个参数的reduce方法实现使用原理

系列文章会搬迁到公众号
后续也会在公众号更新，欢迎关注
微信搜一搜
布衣码农

reduce 归约

(五)函数式编程技巧reduce概念原理

reduce

T reduce(T identity, BinaryOperator<T> accumulator);

参数

- identity：初值，例如：0，1，空字符串
- accumulator：聚合，例如加法

返回值

- 聚合后的结果

reduce的方法有两种使用方式，一种是带初值，一种是没有初值。

reduce的方法有两种使用方式中不同的方法：

java8 源码中的两个 接口 函数式接口 两个参数 之间以此为区别区别的两个不同的方法接口

的区别是不同的一个 Function 的方式

两个不同的接口的 stream 的一个 方法 区别是

Stream的这两种不同方法区别的原因？

两个不同的接口的 stream 的一个 方法 区别是

两个参数

- identity：初值
- accumulator：聚合

函数式接口的概念

(1) 函数式接口的定义方法

```
@FunctionalInterface
T reduce(T identity, BinaryOperator<T> accumulator);

@FunctionalInterface
public interface BinaryOperator<T> extends BiFunction<T,T,T>
```

BinaryOperator 是继Function 的二元函数的函数式接口 是个 函数式接口

(2) 两个 (T, T, T) 入参的二元函数的函数方法

函数式接口的方法为 apply

```
R apply(T1 t1, T2 t2);
```

R apply(T1 t1, T2 t2);

R 是函数的返回值 T1 是函数的第一个参数 T2 是函数的第二个参数

函数式接口的方法 是函数式接口的核心 是 apply 的 方法

apply 的 参数 T1 T2 是 函数的 入参 R 是 函数的 返回值

```
List<Integer> parsed = Arrays.asList(1, 2, 3, 4, 5);

System.out.println(parsed.stream().reduce(0, (a,b) -> a + b));

BinaryOperator<Integer> f = (a, b) -> a + b;
int[] ints = {1, 2, 3};
int result = f.apply(ints[0], ints[1]);
result = f.apply(result, ints[2]);
System.out.println(result);

D:\ApplicationFile\program\jdk1.8\bin\java
-Dfile.encoding=UTF-8 -classpath
D:\ideaWorkspace\Test_Stream\out\production\Test_Stream
Exception in thread "main" java.lang.ClassCastException: class [Ljava.lang.Integer; cannot be cast to class java.lang.Integer ([Ljava.lang.Integer; and java.lang.Integer are in module java.base of loader 'bootstrap')
        at com.test.Test_Stream.lambda$main$0(Test_Stream.java:15)
        at com.test.Test_Stream$$Lambda$1/0x0000000800000840.apply(Unknown Source)
        at java.base/java.util.stream.ReduceOps$ReduceOp$1.makeSink(ReduceOps.java:105)
        at java.base/java.util.stream.AbstractPipeline.wrapAndCopyInto(AbstractPipeline.java:249)
        at java.base/java.util.stream.ReduceOps$ReduceOp.evaluateSequential(ReduceOps.java:708)
        at java.base/java.util.stream.AbstractPipeline.evaluate(AbstractPipeline.java:234)
        at java.base/java.util.stream.ReferencePipeline.reduce(ReferencePipeline.java:479)
        at com.test.Test_Stream.main(Test_Stream.java:15)
Process finished with exit code 1
```

BinaryOperator<T> 是 BiFunction<T,T,T> 的函数式接口

BiFunction<T,T,T> 是 Function 的函数式接口

Function 是函数式接口的核心方法 是一个抽象 方法

**函数式接口的核心方法 是函数式接口的核心 是 apply 的