一、新特性

介绍: Java8可谓是Java语言历史上变化最大的一个版本,其承诺要调整java编程想着函数式风格迈进,这有助于编写出更为简洁、表达力更强,并且在很多情况下能够利用并行硬件的代码。

- Java 8新特性介绍
- Lambda表达式介绍
- 使用Lambda表达式代替匿名内部类
- Lambda表达式的作用
- 外部迭代与内部迭代
- Java Lambda表达式语法详解
- 函数式接口详解
- 传递值与传递行为
- Stream深度解析
- Stream API详解
- 串行流与并行流
- Stream构成
- Stream源生成方式
- Stream操作类型
- Stream转换
- Optional详解
- 默认方法详解
- 方法与构造方法引用
- Predicate接□详解
- Function接口详解
- Consumer接口剖析
- Filter介绍
- Map-Reduce讲解、中间操作与终止操作
- 新的Date API分析

1.1 lambda的基本表达式

```
(param1, param2, param3) -> {
}
```

list.fore

m forEach(Consumer<? super Integer> action)

forEach有个参数Consumer<? super Integer>表示泛型,表示Consumer当前这个参数要么是Integer,要么是Integer的父类

```
list.forEach(new Consumer<Integer>() {
    @Override
    public void accept(Integer integer) {
        System.out.println(integer);
    }
});
```

1.2 函数式接口

- 1、如果一个接口只有一个抽象方法,那么该接口就是一个函数式接口
- 2、如果我们在某个接口上声明了FunctionalInterface,那么编译器就会按照函数式接口要求该接口,不满足条件的话,编译器会给报错
- 3、如果某个几口只有一个抽象方法,但是我们没有给该接口声明FunctionalInterface,那么依旧将该接口当作函数式接口

```
@FunctionalInterface
public interface MyInterface {
    void test();
    String toString();
```

4、注:如果一个接口声明类一个抽象方法(父类的方法),函数式接口的抽象方法个数不增加。本质是接口的实现类也会继承java.lang.Object。

1.3 lambda和函数式接口

lambda针对的是函数式接口。同理:本意就是实现接口里面的哪一个抽象方法

1.4 为何需要lambda表达式

为何需要Lambda表达式

- 在Java中,我们无法将函数作为参数传递给一个 方法,也无法声明返回一个函数的方法
- 在JavaScript中,函数参数是一个函数,返回值是 另一个函数的情况是非常常见的;JavaScript是一 门非常典型的函数式语言

Lambda表达式作用

- Lambda表达式为Java添加了缺失的函数式编程特性,使我们能将函数当做一等公民看待
- 在将函数作为一等公民的语言中,Lambda表达式的类型是函数。但在Java中,Lambda表达式是对象,他们必须依附于一类特别的对象类型——函数式接口(functional interface)

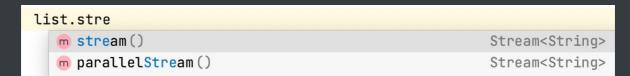
1.5 lambad表达式深入和流初步

(从java8开始接口里面可以包含方法的实现: Default Method, 也可以包含静态方法static method (可以有自己的实现))

(如果一个接口只有一个抽象方法,那么该接口就是一个函数式接口)

```
TheInterface1 t1 = () -> {};
TheInterface2 t2 = () -> {};
System.out.println(t1.getClass().getInterfaces()[0]);
System.out.println(t2.getClass().getInterfaces()[0]);

//lambda表达式依赖于上下文,没有上下文lambda表达式的上下文无从得知的
//函数式接口的抽象方法对lambda来说毫无意义,名字本身对于接口的实现来说还是
很关键的
() -> {};
```



如上: stream和parallelStream面向的领域是不同的

- stream是一个串形流,整个操作上由一个单线程完成的
- parallelStream上并行流,是由多线程完成的