**Java8新特性**

# 一、简介

## 1.主要内容

（1）Lambda表达式

（2）函数式接口

（3）方法引用与构造器引用

（4）Stream API

（5）接口中的默认方法与静态方法

（6）新日期时间API

（7）其他特性

## 2.特点

（1）速度快

HashMap引入红黑树等

jvm 永久区 变为 MetaSpace

（2）代码更少

（3）强大的Stream API

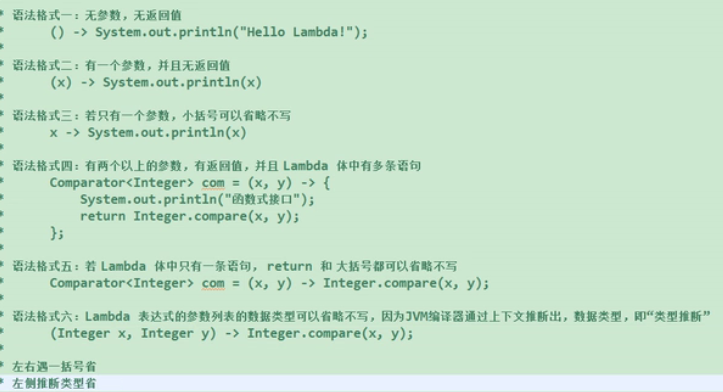
（4）便于并行

（5）最大化减少空指针异常 Optional

# 二、Lambda表达式

简化了函数式接口实现其内部方法的过程

## 1.语法格式



## 2.函数式接口的定义

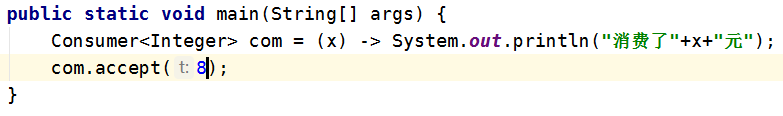
使用注解@FunctionalInterface修饰接口，此接口中只能有一个方法



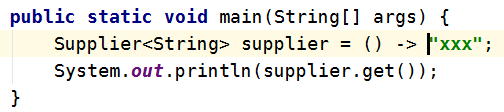
# 三、函数式接口

## 1.内置四大核心函数式接口

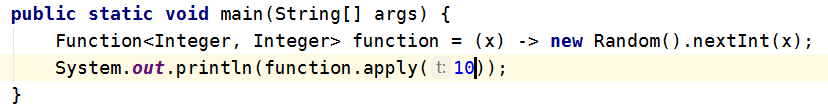
（1）Consumer<T> void accept(T t)



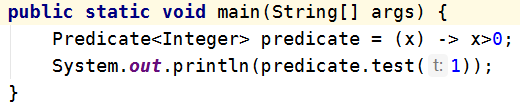
（2）Supplier<T> T get()



（3）Function<T,R> R apply(T t)



（4）Predicate<T> boolean test(T t)

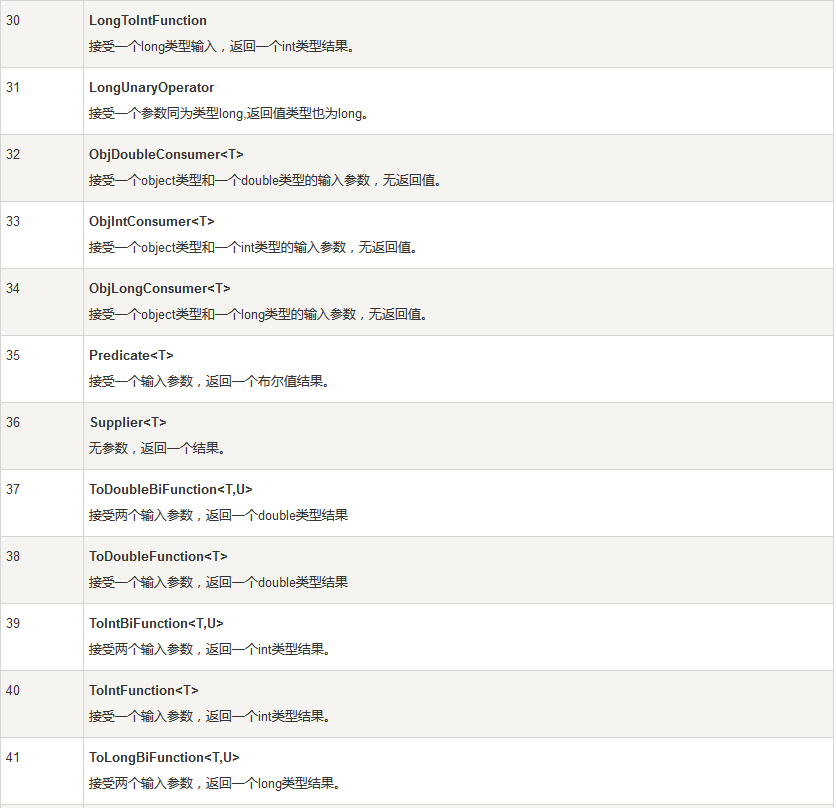


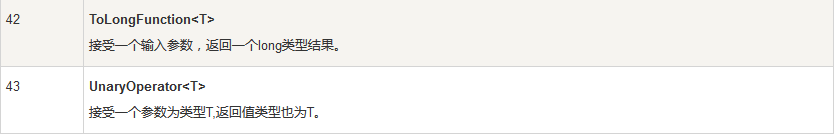
## 2.函数式接口总结







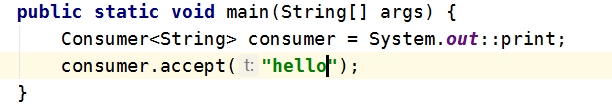




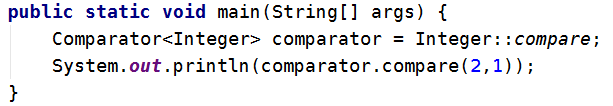
# 四、方法引用与构造器引用

## 1. Lambda体中内容已经有方法实现，使用“方法引用”

（1）对象::实例方法名

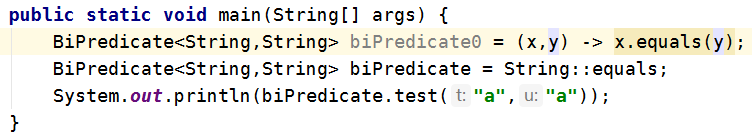


（2）类::静态方法名

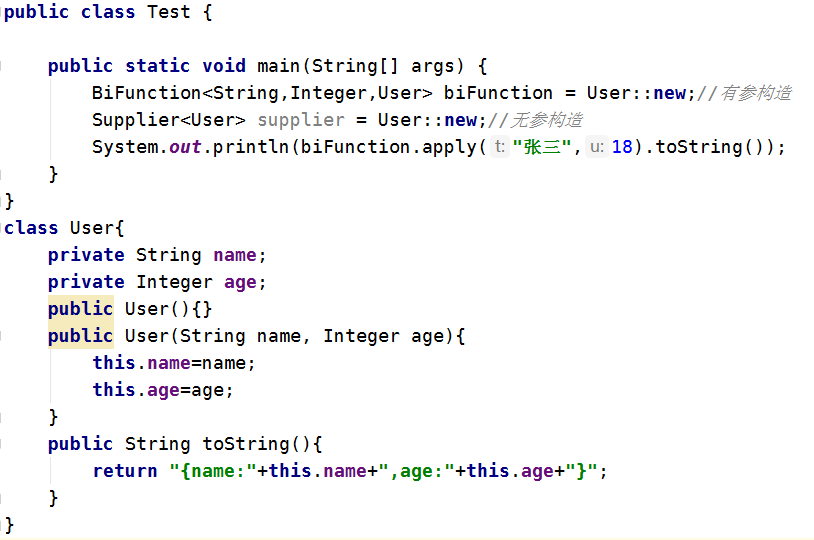


（3）类::实例方法名

实例是一个参数



## 2.构造器引用



# 五、Stream API

## 1.创建

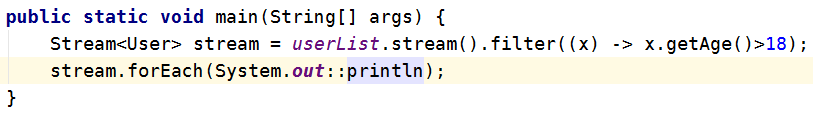
创建Stream的四种方式：



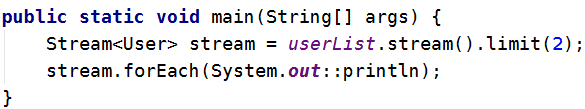
## 2.中间操作

### （1）筛选与切片

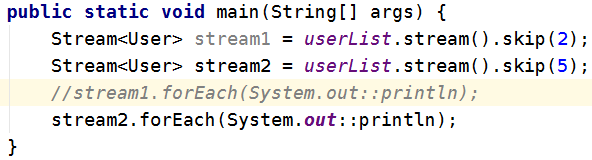
<1>filter 接收Lambda，从流中排出某些元素



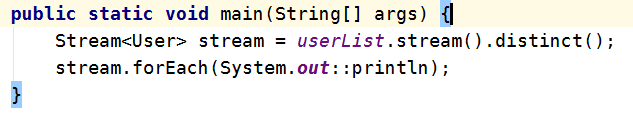
<2>limit 截断流，使其元素不超过给定数量



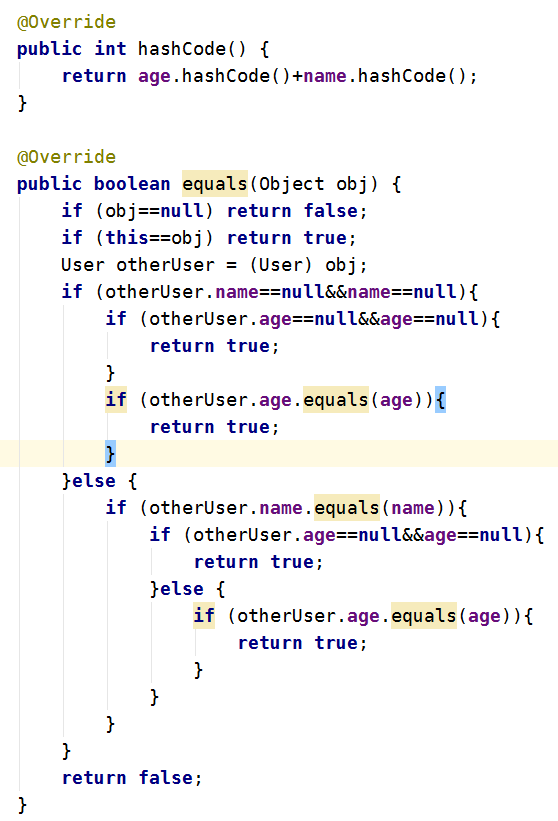
<3>skip 跳过元素，返回一个扔掉了前n个元素的流，若元素不足n个，则返回一个空流



<4>distinct 去重筛选，通过流所生成的元素的hashCode()和equals() 去除重复元素

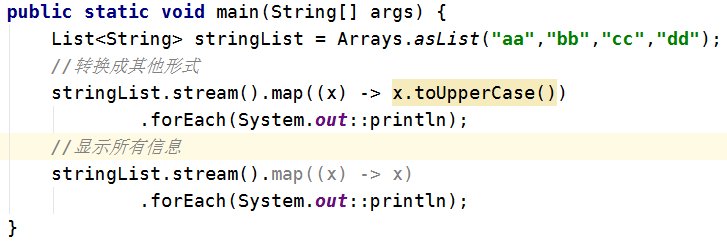


需要重写hashCode和equals，判断二者是否重发，是先判断codeCode是否相等再判断equals

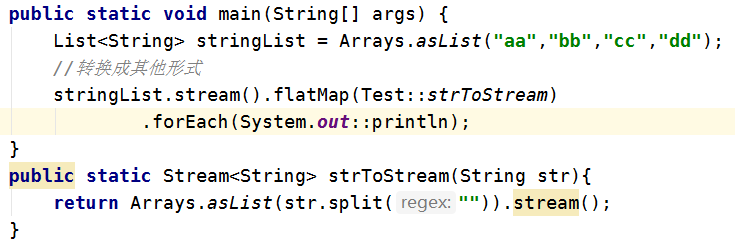


### （2）映射

<1>map 接收Lambda，将元素转换成其他形式或提取信息。接收一个函数作为参数，该函数会被应用到每个元素上，并将其映射成一个新元素



<2>flatMap 接收一个函数作为参数，将流中的每个值都换成另一个流，然后把所有流连接成一个流

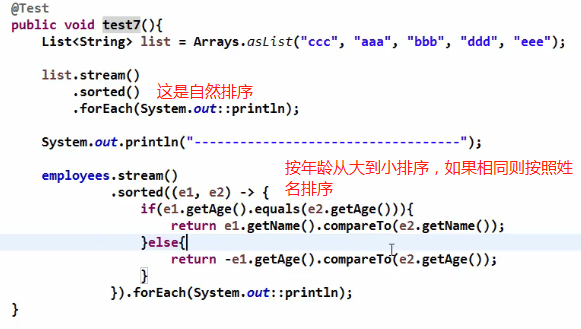


其实，就是对List中的所有对象进行转换成流，再该流上做操作，然后在合并这些流。

### （3）排序

<1>sorted 自然排序

字符串自然排序 是按照String实现的Comparable<String>中的compareTo方法排序



<2>sorted(Comparator com) 定制排序

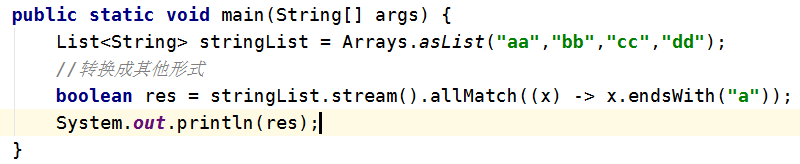
### （4）去重

https://blog.csdn.net/haiyoung/article/details/80934467

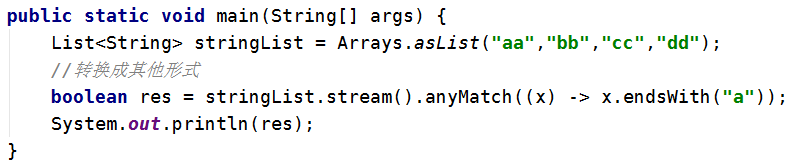
## 3.终止操作

### （1）查找与匹配

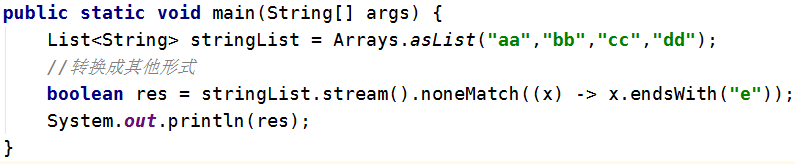
<1>allMatch 检查是否匹配所有元素



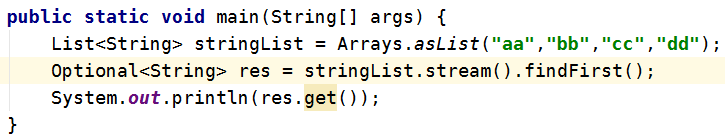
<2>anyMath 检查是否至少匹配一个元素



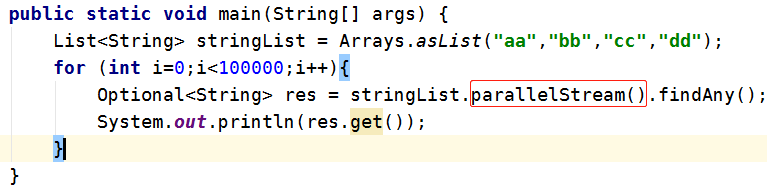
<3>noneMatch 检查是否没有匹配所有元素



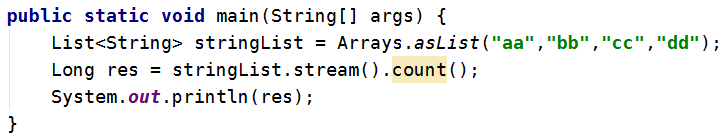
<4>findFirst 返回第一个元素



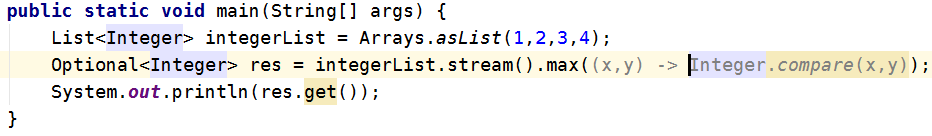
<5>findAny 返回当前流中任意元素



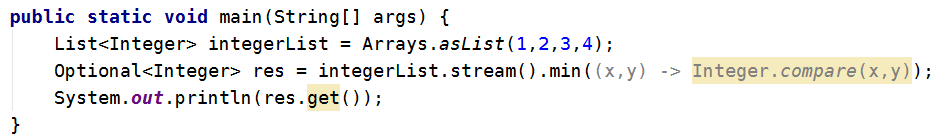
<6>count 返回当前流中的总个数



<7>max 返回流中最大值



<8>min 返回流中最小值

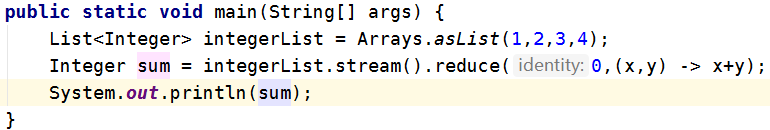


### （2）归约与收集

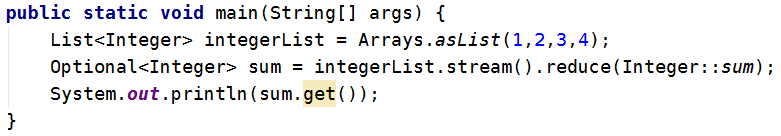
<1>reduce(T identity, BinaryOperator) 或 reduce(BinaryOperator)

可以将流中元素反复结合起来，得到一个值

eg：两个参数，第一个是起始位置



eg：一个参数

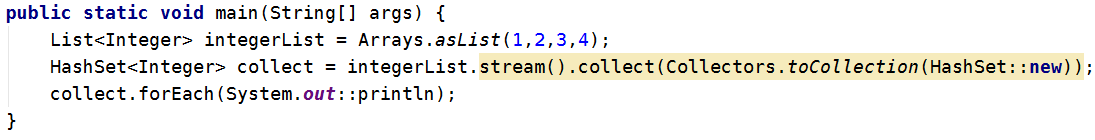


map-reduce模式，Google用其搜索

<2>collect

将流转化为其他形式，接收一个Collector接口的实现，用于给Stream中元素做汇总的方法

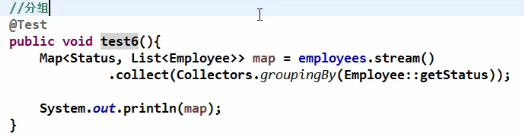
eg：转化成自定义的集合



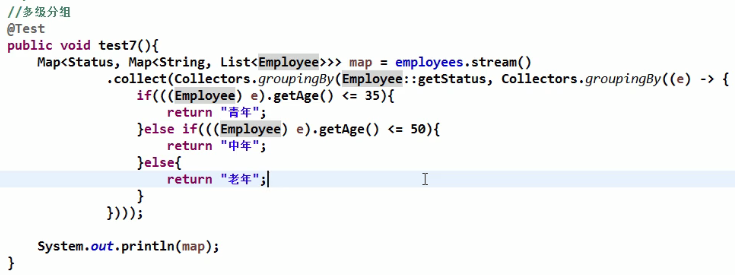
eg：最大，最小，总数，平均值，总和



eg：分组



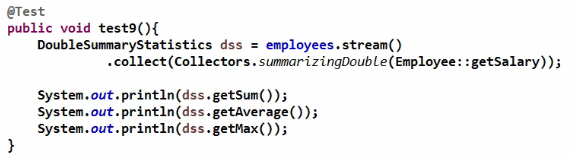
eg：多级分组，先按状态分，再按年龄分



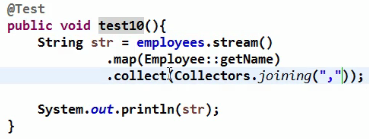
eg：分区，工资大于8000一个区，其他的在另一个区



eg：生成DoubleSummaryStatistics对象来计算平均值，总和等



eg：连接字符串



### （3）串行流和并行流

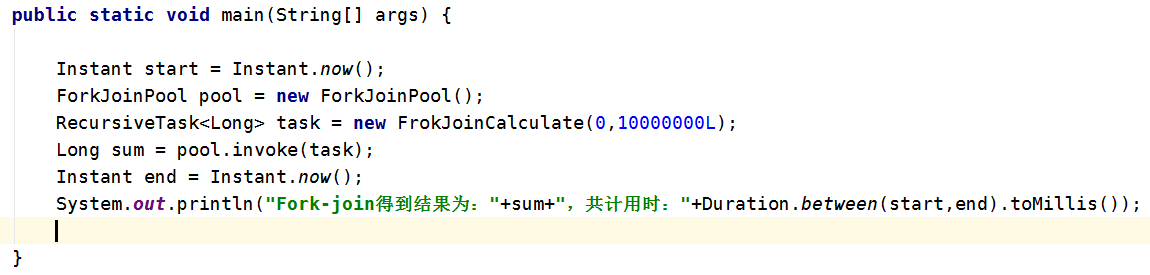
<1>fork-join框架

将一个大任务拆分成无数个小任务，计算完成后合并起来

窃取模式：自己的线程队列（双端队列）执行完成后，随机去其他线程队列偷一个任务去执行

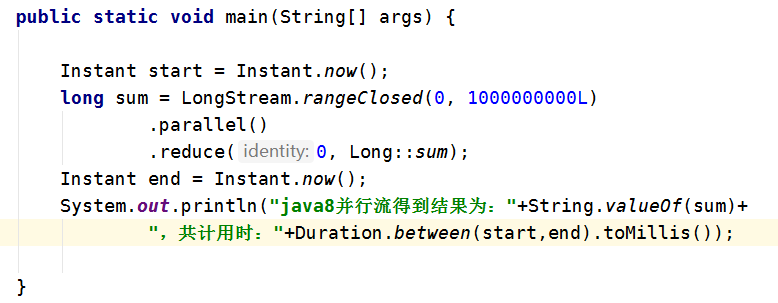
例子：完成对从start到end的累加





<2>java8 并行流

parallel() 切换到并行流  
sequential() 切换到串行流



# 六、Optional判空

## 1.创建实例

Optional <T> Optional.of(T t) ：创建一个Option实例

Optional <T> Optional.empty() ：创建一个空的实例

Optional <T> Optional.ofNullable(T t) ：若t不为null，则创建一个实例；否则，创建一个空的实例

## 2.判断实例是否为null

bool isPresent() ：判断包含的对象是否为null

## 3.获取其包含的对象

T get() ：获取其包含的对象

T orElse(T t) ：获取其包含的对象，如果该对象为null，则返回传入的对象

T orElseGet(Supplier s) ：获取其包含的对象，如果该对象为null，则返回Supplier返回的对象

Optional<U> map(Function f) ：返回Function返回的被Optional包裹的对象（自动包裹）

Optional<U> flatMap(Function f) ：返回Function返回的被Optional包裹的对象（需要手动包裹）

# 七、接口中的默认方法与静态方法

## 1.接口中的默认方法

修饰符 default

eg： default String getName(){ return "Sam"; }

当实现接口中的默认方法和继承类中的方法命名冲突时，使用继承类中的方法

当实现两个接口中的默认方法命名冲突时，实现类需要声明实现的是哪个方法

## 2.接口中的静态方法

public static int getAge(){ return 18; }

# 八、新日期时间API

**1.旧的**

对日期加减操作的Calendar在java.util下

对日期格式化操作的SimpleDateFormate在java.text下

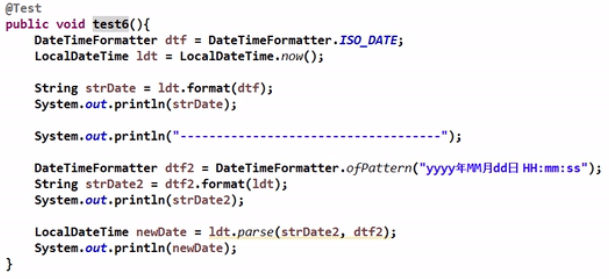
**2.新的**

（1）在java.time下



（2）java.time.chrono 对其他国家时间的特殊化处理

（3）java.time.format 对日期格式化处理

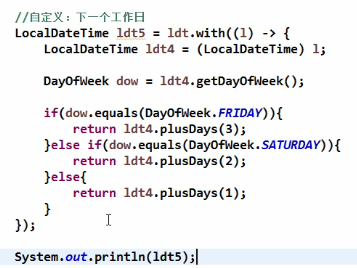


（4）java.time.temporal 加减时间

TemporalAdjuster 时间校正器



eg：获取下一个工作日



（5）java.time.zone 时区



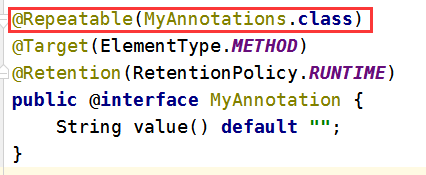
**3.线程安全**

对日期格式化处理线程安全

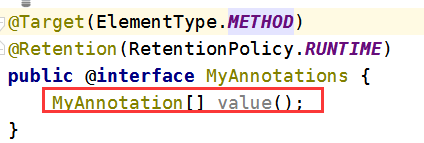
# 九、其他特性

## 1.可重复注解

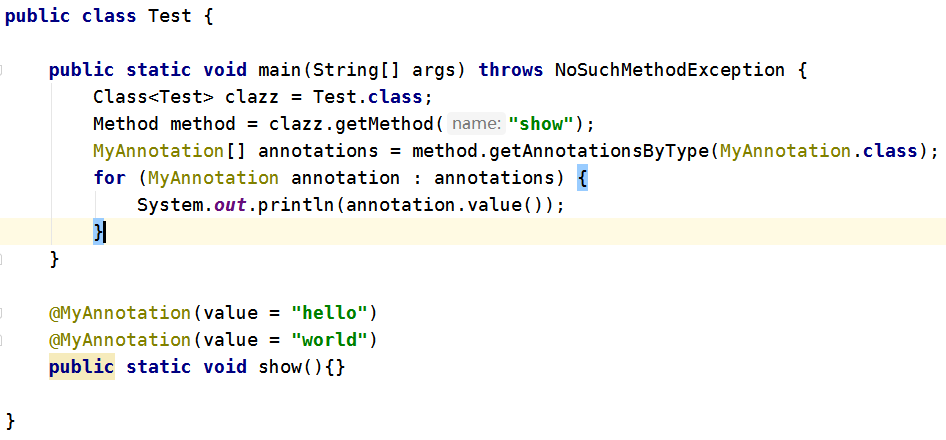
（1）先自定义一个注解，标识@Repeatable，指定容器注解



（2）定义容器注解

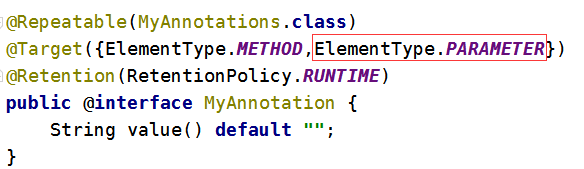


（3）实现多重注解



## 2.类型注解

（1）定义



（2）使用

