Lambda表达式：允许把函数作为一个方法的参数。

方法引用：可以直接引用已有java类或对象的方法或构造器。

默认方法：默认方法就是在一个接口里面有了一个实现方法。

新工具：新的编译工具，如：Nashorm引擎jjs、类依赖分析器jdeps

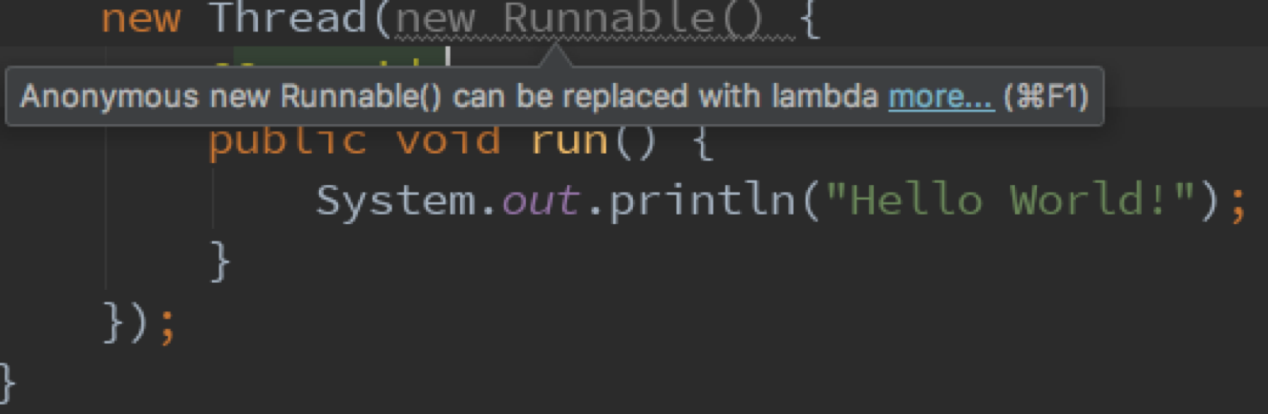
Stream API：新添加的Stream API(java.uitl.stream)把真正的函数式编程风格引入到java中。

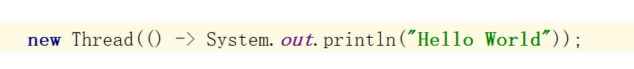
Date Time API：加强对日期与时间的处理

Optional类：Optional类已经成为java8类库的一部分，用来解决空指针异常。

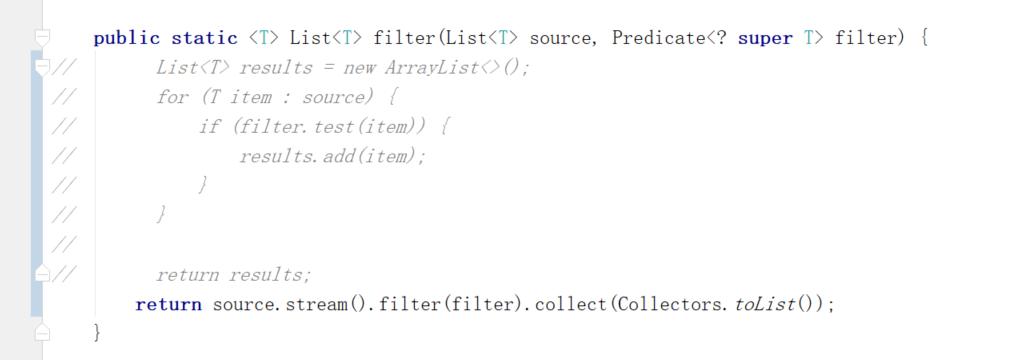
Nashorn，JavaScript引擎：允许我们在JVM上运行特定的JavaScript应用

1. **Lambda表达式**：
2. **能够接收Lambda表达式的参数类型，是一个只包含一个方法的接口**。只包含一个方法的接口称之为“**函数接口**”。



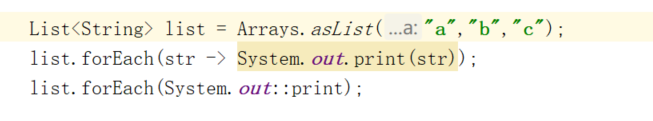


上图第一个是常规写法，第二个是Lambda表达式写法。



以上是普通写法和优化写法

1. 基本分为3种情况：无参数，无返回值；有参数，无返回值；有参数，有返回值。
2. 相关的函数接口：function（函数，接收一个泛型，返回一个泛型，apply）、operator（算子,apply）、predicate（谓词函数，接收一个泛型，返回一个boolean值,test）、consumer（接收一个泛型， 不需要返回值,accept）、supplier（声明获取一个泛型值,get）。
3. **方法引用**：

**只需要使用方法的名字，而具体调用交给函数接口，需要和Lambda表达式配合使。**

以上两种写法是等价的。

构造方法引用：Class ：：new，调用默认构造器

类静态方法引用：Class ：：static\_method

类普通方法：Class ：：method（方法不能带参数）

实例方法引用：instance ：：method

1. **默认方法**：
2. 接口可以有默认方法，在进行接口扩展时，提供默认方法，原接口的实现类就不用进行修改。
3. 冲突规则：a、接口冲突：如果一个类同时实现了具有相同方法签名的接口，则该类必须覆盖该抽象方法；b、超类冲突：超类和接口提供相同方法签名的方法，此时子类要是不覆盖，则默认调用超类方法。
4. **Stream API**：
5. Stream是数据渠道，用于操作数据源（集合、数组）所生成的元素序列。**集合讲的是数据，流讲的是计算！**stream自己不会存储元素，不会改变源对象，会返回一个持有结果的新stream。stream其操作时延迟执行的，这意味着他们会等到需要结果的时候才执行。
6. Stream的操作三个步骤：**创建stream**（一个数据源，获取一个流）、**中间操作**（一个中间操作链）、**终止操作**（一个终止操作，执行中间操作链，并产生结果）
7. 创建stream：
   1. 可以通过Collection系列集合提供的串行流stream（）或并行流慎用parallelStream（）
   2. 通过Arrays中的静态方法stream（）获取数组流
   3. 通过stream类中的静态方法of（）
   4. 创建无限流，Stream.iterate（）迭代、Steam.generate（）生成
8. 什么时候用stream或者parallelStream，考虑以下三点：
9. 是否需要并行？
10. 任务之间是否独立？是否会引起竞态条件？
11. 结果是否取决于任务的调用顺序？
12. 中间操作：**多个中间操作可以连接起来形成一个流水线，除非流水线上触发终止操作，否则中间操作不会执行任何的处理，而在终止操作时一次性全部处理，称为惰性求值。**
    1. 筛选与切片：
       1. filter（Predicate）：接收Lambda，从流中排除某些元素
       2. distinct（）：筛选，通过流所生成元素的hashCode（）和equals（）去除重复元素
       3. limit（long）：截断流，使其元素不超过给定数量。
       4. skip（long）：跳过元素，返回一个扔掉前n个元素的流。若流中元素不足n个，则返回一个空流。与limit互补
    2. 映射：
       1. map（Function）：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的元素。
       2. flatMap（Function）：接收一个函数作为参数，将流中的每一个值都转化成另一个流，然后把所有流连接成一个流。
    3. 排序：
       1. sorted（）：产生一个新流，其中按自然顺序排序。
       2. sorted（Comparator）：产生一个新流，其中按比较器顺序排序。
13. 终止操作：**终端操作会从流的流水线生成结果。其结果可以是任何不是流的值**。
    1. 查找与匹配：
       1. allMatch（Predicate）：匹配所有元素
       2. anyMatch（Predicate）：至少匹配一个元素
       3. noneMatch（Predicate）：没有元素匹配
       4. findFisrt（）：返回第一个元素
       5. findAny（）：返回当前流中的任意元素
       6. count（）：返回流中元素的总数
       7. min（Comparator）：返回流中最小元素
       8. max（Comparator）：返回流中最大元素
       9. forEach（Consumer）：内部迭代
    2. 归约：
       1. reduce（T, BinaryOperator)：可以将流中元素反复结合起来，得到一个值。返回T
       2. reduce（BinaryOperator）：可以将流中元素反复结合起来，得到一个值，返回Optional<T>
    3. 收集：collect（Collector）将流转化为其他形式（list、set、map），Collectors实用类提供了很多静态方法，可以方便的创建常见收集器实例。
       1. toList（）：把流中元素收集到List，返回List<T>
       2. toSet（）：把流中元素收集到Set，返回Set<T>
       3. toCollection（）：把流中元素收集到创建的集合中，返回Collection<T>
       4. counting（）：计算流中元素的个数，返回Long
       5. summingInt（）：对流中元素的整数属性求和，返回Integer
       6. averagingInt（）：极端流中整数属性的平均值，返回Double
       7. summarizing（）：收集流中整数属性的统计值，返回IntSummaryStatistics
       8. joining（）：连接流中每个字符串，返回String
       9. maxBy（）：根据比较器选择最大值，返回Optional<T>
       10. minBy（）：根据比较器选择最小值，返回Optional<T>
       11. reducing（）：从一个作为累加器的初始值开始，利用BinaryOperator与流中的元素逐个结合，从而归约成单个值，返回归约类型
       12. collectingAndThen（）：包裹另一个收集器，对其结果转化函数，返回转化函数类型
       13. groupingBy（）：根据某属性值对流分组，属性为K，结果为V，返回Map<K, List<T>>
       14. partitioningBy（）：根据true或false分区，返回Map<Boolean, List<T>>
14. **Date Time API**：解决了以前时间日期线程不安全的问题
15. Instant类表示时间线上的一个点。参考点是标准的java纪元，即1970-01-01。纪元之后的时间是正值，而在此之前的时间是负值。
16. static now（）：返回一个标识当前时间的instant对象
17. getEpochSecond（）：返回自纪元以来经过的秒数
18. getNano（）：返回自上一秒开始以来的纳秒数
19. EPOCH属性：返回标识java纪元的instant实例
20. LocalDate类只包括日期没有时间的部分，也没有时区。localDate是不可变的，因此无法更改，任何返回localDate的方法都返回localDate的新实例。
21. static now（）：返回今天的日期
22. static of（）：从指定年份，月份，日期创建localDate
23. getDayOfMonth，getMonthValue，getYear（）：以int形式返回此localDate的日，月或年
24. getMonth（）：以Month枚举常量返回此localDate的月份
25. plusDays，minusDays（）：给localDate添加或减去指定的天数
26. plusWeeks，minus Weeks（）：给localDate添加或减去指定的星期数
27. plusMonths，minusMonths（）：给localDate添加或减去指定的月数
28. plusYears，minusYears（）：给localDate添加或减去指定的年数
29. isLeapYear（）：检查localDate的年份是否是闰年
30. isAfter，isBefore（）：检查localDate在给定日期之后还是之前
31. lengthOfMonth（）：返回此localDate中月份的天数
32. withDayOfMonth（）：返回此localDate的拷贝，将月份中的某天设置为给定值
33. withMonth（）：返回此localDate的拷贝，其月份置为给定值
34. withYear（）：返回此localDate的拷贝，其年份置为给定值
35. get（）：接收一个TemporalField并返回这个localDate的一部分
36. LocalDateTime类是一个没有时区的日期时间的构建。方法与localDate类似，只不过多了一些时、分、秒的类似操作。
37. Period类基于日期的时间数量构建，用来测量时间总计。
38. between（）：在两个local Date之间创建一个period实例
39. ofDays，ofWeeks，ofMonths，ofYears（）：创建代表给定天/周/月/年的period实例
40. of（）：根据给定的年数，月数和天数创建一个period实例
41. getDays，getMonths，getYears（）：以int形式返回此period的天/月/年
42. isNegative（）：如果此period的三个部分中的任何一个为负数，则返回true，否则返回false
43. isZero（）：如果此period的所有三个部分均为零，则返回true，否则返回false
44. plusDays，minusDays（）：在此period上添加或减去给定的天数
45. plusMonths，minusMonths（）：在此period上添加或减去给定月数
46. plusYears，minusYears（）：在此period上添加或减去给定年数
47. withDays（）：以指定的天数返回此period的拷贝
48. withMonths（）：以指定的月数返回此period的拷贝
49. withYears（）：以指定的年数返回此period的拷贝
50. Duration类是基于时间的持续时间的构建。它与period类似，不同之处在于duration的时间分量为纳秒精度，并考虑了ZonedDateTime实例之间的时区。
51. java日期和时间API也适用于时区。抽象类ZoneId表示一个区域表示符。它有一个名为getAvailableZoneIds的静态方法，它返回所有区域标识符。
52. ZonedDateTime类是一个以时区日期时间的构建，这是一个最终类，时间分量的存储精度为纳秒。方法与localDateTime类似，只不过多时区操作。
53. DateTimeFormatter类用于格式化本地或时区日期时间
54. **Optional类**：这是一个可以为null的容器对象。如果值存在isPresent（）方法会返回true，调用get（）方法会返回该对象。
55. static Optional empty（）：返回空的optional实例
56. boolean equals（Obejict）：判读其他对象是否登陆optional
57. Optional filter（Predicate）：如果值存在，并且这个值匹配给定的predicate，返回一个optional用以描述这个值，否则返回一个空的optional
58. Optioanl flatMap（Function）：如果值存在，返回基于optional包含的映射方法的值，否则返回一个空的optional
59. get（）：如果这个Optional包含这个值，返回这个值，否则抛出异常：NoSuchElementException
60. hashCode（）：返回存在值的哈希码，如果值不存在返回0
61. ifPresent（Consumer）：如果值存在使用该值调用consumer方法，否则不做任何事
62. ifPresent（）：如果值存在返回true，否则返回false
63. map（Function）：如果有值，则对其执行调用映射函数得到返回值。如果返回值不为null，则创建包含映射返回值的optional作为map方法返回值，否则返回空Optional
64. static of（T）：返回一个指定非null值的optional
65. static ofNullable（T）：如果非空，返回optional描述的指定值，否则返回空的Optional
66. orElse（T other）：如果存在该值，返回值，否则返回other
67. orElseGet（T other）：如果存在该值，返回值，否则触发other，并返回other调用的结果
68. orElseThrow（Supplier）：如果该值存在，返回值，否则抛出supplier继承的异常
69. toString（）：返回一个Optional的非空字符串