Stream：对指定集合进行复杂的查找、过滤、映射数据等操作，可以串行也可以并行

1. 注意：
   1. Stream本身不会存储元素
   2. Stream不会改变源对象，会返回一个持有结果的新Stream
   3. Stream操作具有延迟，也就是需要结果的时候才执行
2. 三个步骤：
   1. 创建Stream：获取一个数据源（集合、数组），获取一个流
      1. 使用Collection接口中的方法
         1. default Stream<E> stream() : 返回一个顺序流
         2. default Stream<E> parallelStream() : 返回一个并行流
      2. 使用数组创建流，具有多个重载形式
         1. static <T> Stream<T> stream(T[] array): 返回一个流
         2. public static IntStream stream(int[] array)
         3. public static LongStream stream(long[] array)
         4. public static DoubleStream stream(double[] array)
      3. 使用值创建流，使用静态方法Stream.of()，可以接受任意数量的参数
         1. public static<T> Stream<T> of(T... values) : 返回一个流
      4. 使用函数创建流，无限流，使用静态方法Stream.iterate()和Stream.generate()
         1. public static<T> Stream<T> iterate(final T seed, final UnaryOperator<T> f) 迭代
         2. public static<T> Stream<T> generate(Supplier<T> s) 生成
   2. 中间操作：一个中间操作链，对数据源的数据处理
      1. 筛选与切片
         1. filter(Predicate p) 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 maxSize) 截断流，使其元素不超过给定数量。
         4. skip(long n) 跳过元素，返回一个扔掉了前 n 个元素的流。若流中元素不足 n 个，则返回一个空流。与 limit(n) 互补
      2. 映射
         1. map(Function f) 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. mapToDouble(ToDoubleFunction f) 接收一个函数作为参数，该函数会被应用到每个元素上，产生一个新的 DoubleStream。
         3. mapToInt(ToIntFunction f) 接收一个函数作为参数，该函数会被应用到每个元素上，产生一个新的 IntStream。
         4. mapToLong(ToLongFunction f) 接收一个函数作为参数，该函数会被应用到每个元素上，产生一个新的 LongStream。
         5. flatMap(Function f) 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
      3. 排序：
         1. sorted() 产生一个新流，其中按自然顺序排序
         2. sorted(Comparator comp) 产生一个新流，其中按比较器顺序排序
   3. 终止操作：终止操作会执行中间操作链，并产生结果
      1. 查找与匹配
         1. allMatch(Predicate p) 检查是否匹配所有元素
         2. anyMatch(Predicate p) 检查是否至少匹配一个元素
         3. noneMatch(Predicate p) 检查是否没有匹配所有元素
         4. findFirst() 返回第一个元素
         5. findAny() 返回当前流中的任意元素，多个挑一个
         6. count() 返回流中元素总数
         7. max(Comparator c) 返回流中最大值
         8. min(Comparator c) 返回流中最小值
         9. forEach(Consumer c) 内部迭代(使用 Collection 接口需要用户去做迭代，称为外部迭代。相反，Stream API 使用内部迭代——它帮你把迭代做了)
      2. 归约
         1. reduce(T iden, BinaryOperator b) 可以将流中元素反复结合起来，得到一个值。返回 T
         2. reduce(BinaryOperator b) 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
         1. collect(Collector c) 将流转换为其他形式。接收一个 Collector接口的实现，用于给Stream中元素做汇总的方法
            1. Collector 接口中方法的实现决定了如何对流执行收集操作(如收集到 List、Set、Map)。但是 Collectors 实用类提供了很多静态方法，可以方便地创建常见收集器实例。





1. 串行流与并行流
   1. 并行流：把一个内容分成多个数据块，并用不同的线程分别处理每个数据块的流
   2. Stream API通过parallel()与sequential()进行并串切换
   3. 底层使用Fork/Join框架