heros

.stream()

.filter(h -> h.hp > 100 && h.damage < 50)

.forEach(h -> System.out.println(h.name));

要了解聚合操作，首先要建立**Stream**和**管道**的概念  
**Stream**和Collection结构化的数据不一样，Stream是一系列的元素，就像是生产线上的罐头一样，一串串的出来。  
**管道**指的是一系列的聚合操作。  
  
管道又分3个部分  
**管道源**：在这个例子里，源是一个List  
**中间操作**： 每个中间操作，又会返回一个Stream，比如.filter()又返回一个Stream, 中间操作是“懒”操作，并不会真正进行遍历。  
**结束操作**：当这个操作执行后，流就被使用“光”了，无法再被操作。所以这必定是流的最后一个操作。 结束操作不会返回Stream，但是会返回int、float、String、 Collection或者像forEach，什么都不返回, 结束操作才进行真正的遍历行为，在遍历的时候，才会去进行中间操作的相关判断  
  
**注：** 这个Stream和I/O章节的InputStream,OutputStream是不一样的概念。

把Collection切换成管道源很简单，调用stream()就行了。

heros.stream()  
但是数组却没有stream()方法，需要使用

Arrays.stream(hs)

或者

Stream.of(hs)

每个中间操作，又会返回一个Stream，比如.filter()又返回一个Stream, 中间操作是“懒”操作，并不会真正进行遍历。  
中间操作比较多，主要分两类  
对元素进行筛选 和 转换为其他形式的流  
****对元素进行筛选：****  
filter 匹配  
distinct 去除重复(根据equals判断)  
sorted 自然排序  
sorted(Comparator<T>) 指定排序  
limit 保留  
skip 忽略  
****转换为其他形式的流****  
mapToDouble 转换为double的流  
map 转换为任意类型的流

当进行结束操作后，流就被使用“光”了，无法再被操作。所以这必定是流的最后一个操作。 结束操作不会返回Stream，但是会返回int、float、String、 Collection或者像forEach，什么都不返回,。  
结束操作才真正进行遍历行为，前面的中间操作也在这个时候，才真正的执行。  
常见结束操作如下：  
****forEach()**** 遍历每个元素  
****toArray()**** 转换为数组  
****min(Comparator<T>)**** 取最小的元素  
****max(Comparator<T>)**** 取最大的元素  
****count()**** 总数  
****findFirst()**** 第一个元素