一个非常常见的数据处理套路就是从某些对象中选择信息。比如在SQL里，可以从表中选择一列。Stream API也通过map和flatMap方法提供了类似的工具。

#### 1、对流中每一个元素应用函数

流支持map方法，接受一个函数作为参数。这个函数会被应用到每个元素上，并将其映射成一个新的元素（使用映射一词，是因为它和转换类似，但其中的细微差别在于它是“创建一个新版本”而不是去“修改”）。

.map(Person::getName)

因为getName方法返回一个String，所以map方法输出的流的类型就是Stream<String>。

.map(String::length)

#### 2、流的扁平化

对于一张单 词表，如何返回一张列表，列出里面各不相同的字符呢?例如，给定单词列表 ["Hello","World"]，你想要返回列表["H","e","l","o","W","r","d"]。你可能会认为这很容易，你可以把每个单词映射成一张字符表，然后调用distinct来过滤 重复的字符。第一个版本可能是这样的：

List<String> words = new ArrayList<>();

words.add("Hello");  
words.add("World");  
System.*out*.println(words);  
List<String[]> list = words.stream()  
 .map(w -> w.split(""))  
 .distinct()  
 .collect(*toList*());

这个方法的问题在于，传递给map方法的Lambda为每个单词返回一个String[](String列表)。因此，map返回的流实际上是Stream<String[]>类型的。真正想要的是用Stream<String>来表示一个字符流。这时使用flatMap来解决这个问题。

List<String> words = new ArrayList<>();

words.add("Hello");  
words.add("World");  
List<String> uniqueCharacters = words.stream()  
 .map(w -> w.split(""))  
 .flatMap(Arrays::*stream*)// 将各个生成流扁平化为单个流  
 .distinct()  
 .collect(Collectors.*toList*());

所有使用map(Arrays::Stream)时生成的单个流都被合并起来，即扁平化为一个流。flatMap方法就是让你把一个流中的每个值都换成另一个流，然后把所有的流连接起来成为一个流。

