流与文件

代码文件位于 `streamprac\src\com\Jkong` 目录

Task 1　流API

1. 运行给出的代码会得到 类名@地址名 的结果。这是因为Stream类中并没有重写 `toString()` 方法，所以直接打印Stream对象时就会直接打印 类名@地址值，也就是Object中 `toString()` 的输出。

2. 如果想要打印这个Stream的元素，有两种解决方案，我在test目录的test_one中写了出来，分别是 `collect(Collectors.toList())` 和简单的 `toList()` 。本来我疑惑这两者功能是否发生了重叠，查阅资料才得知，后者是java 16及以后才能使用的方法，而且生成的是一个不可变的列表，而前者生成的是一个可变的普通列表。与此同时，`Collectors.toUnmodifiableList()` 方法也可以生成一个不可变的列表。

3. 题目给出的代码我贴在了given_test文件中。这堆操作能堆叠的原理是，中间操作 `.map` 和 `.sorted` 都满足一个条件，即输入的是一个Stream，返回的也是一个Stream。这样，整个语句就像一条流水线。

Lambda

1. 还好这几天在B站大学了解了匿名内部类。现在我知道，Lambda表达式可以视作匿名内部类的一种简化表达。但是相比于普通的匿名内部类，Lambda表达式只能实现接口，而不能继承类，而且相应接口内必须有且只有一个抽象方法。

2. 以上面given_test中的代码为例，`map()` 方法和 `sorted()` 方法都要求传入形参（前者要求传入 `Function`，后者要求的是Comparator的实现类）。如果新建一个类并传入其中的方法，未免太费时，所以就可以使用lambda表达式简化。比如 `i -> i * i` 的意思是，创建一个类，它有某个方法，对于输入的形参i，它会返回i * i。`map()` 方法随后会将每个元素都替换成它的平方。又如 `(x,y) -> y - x` ，它的形参则是x和y，返回的是y-x的结果。`sorted()` 方法根据其返回的值决定如何排序。所以这里的意思是按从大到小顺序排列。

歌曲分类进阶

代码都放在了application目录内，**截图名为song_list_result**

1. 只需要使用 `filter()` 方法，检查genre为"Rock"的Song对象并保留，然后再转成一个新List就可以了。

2. 使用 `map()` 方法，将每一个Song对象都替换为它们对应的genre字符串，然后再使用 `distinct()` 方法，去除stream中重复的元素，转换成新List。

虽然这些Stream方法的用法我大致了解，但其底层实现对我来说还是比较复杂。翻源码也很难看得懂，特别是其中存在的复杂的继承与实现关系。比如Stream和IntStream等等

Task 2　串行化

很容易发现，Serializable接口里不存在任何方法，是一个“标记性接口”，它的作用是在对象序列化时防止抛出IOException。在菜鸟教程上了解到，序列化是将对象“存储”起来的方法，它在对象的“传输”上十分有用。

这里用到的代码存放在application_dev目录下。

Song和Songs类都复用，创建新的SongIO类，创建程序主入口。

1. 首先键盘录入想要存的歌曲的序号，然后调用 `SongOutput()` 方法，将歌曲列表中对应序号的Song对象进行序列化，然后通过 `FileOutputStream` 输出到SongOutput.txt中。
2. 随后调用 `SongInput()` 方法，将文件里存放的Song对象反序列化，用变量aimSong进行接收，然后打印出来。

这样进行的是**单次的**输入输出操作，于是我对SongIO进行了功能更改，让它直接将列表中所有的歌曲都写入到文件中，然后再询问要查询的歌曲名字，并输出读入的歌曲的信息。这没什么困难，只需遍历歌曲列表并调用 `SongOutput()` 方法即可。

本来我以为需要加代码，让大小写都能识别，但是测试发现本来文件名查找就不分大小写。

截图名为song_io_result

文件I/O

我记得这个在第七题涉及过，那我干脆把那里的代码改进一下贴到这里。

代码放在file_io目录，名为NumInitializer，会在streamprac目录的num.txt中写入数字（跟歌曲文件混在一起可能不好找）。

截图名为file_io