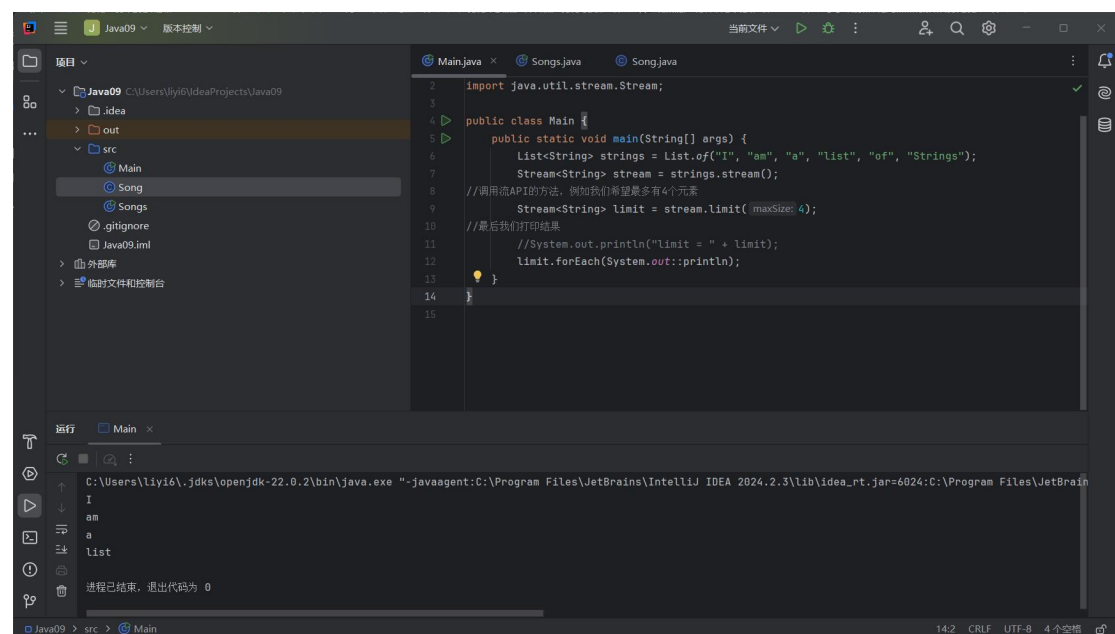


Task1

并不打印流中元素的原因：

使用 `System.out.println("limit = " + limit);` 这个指令仅仅只是打印出了这个流整体的一个字符串表达形式（感觉跟我在 Java-03 中通过调取哈希码的方式来看两个对象是否相同是一个道理，但可能这个表达不咋准确），根据输出来看，里面有流的类和对应的哈希码。

要打印流中元素的话，我要使用流的一个终端操作 `forEach` 内调用 `System.out::println` 来实现。实现的截图：



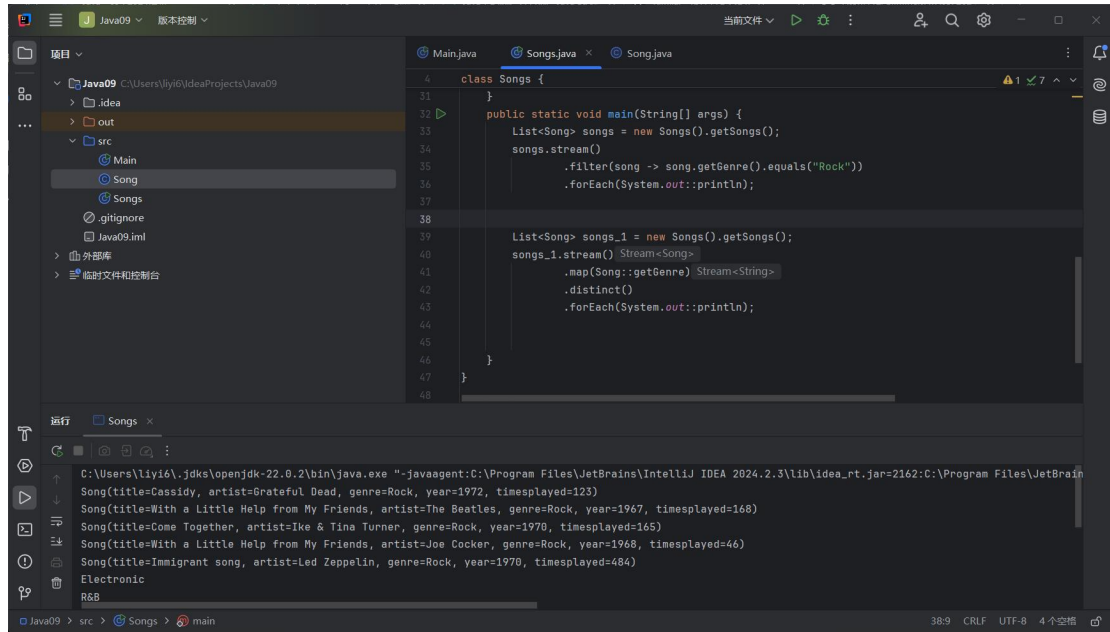
流的使用规则

1. 创建一个流：可以选择从集合、数组、流创建器（`Stream.Builder`）、文件等创建流。
2. 中间操作：这里包含了操作的堆叠，所以我就把两个问题连在一起回答，中间操作有类似示例代码中很多种，可以堆叠使用，当我进行中间操作时会返回一个新的流，直到我进行终端操作为止，而且中间操作不会立即被执行，只有我进行终端操作时它才会处理数据。
3. 终端操作：我上面的 `forEach` 就是其中的一种，进行了这种操作后，流中的数据将会根据中间操作来处理并生成结果，同时这个流就被 KO 了，不能再对其进行另外的中间操作。（小声：创建流什么的，用下面的进阶挑战来顺路做了吧(^_^)

进阶挑战——应用流 API

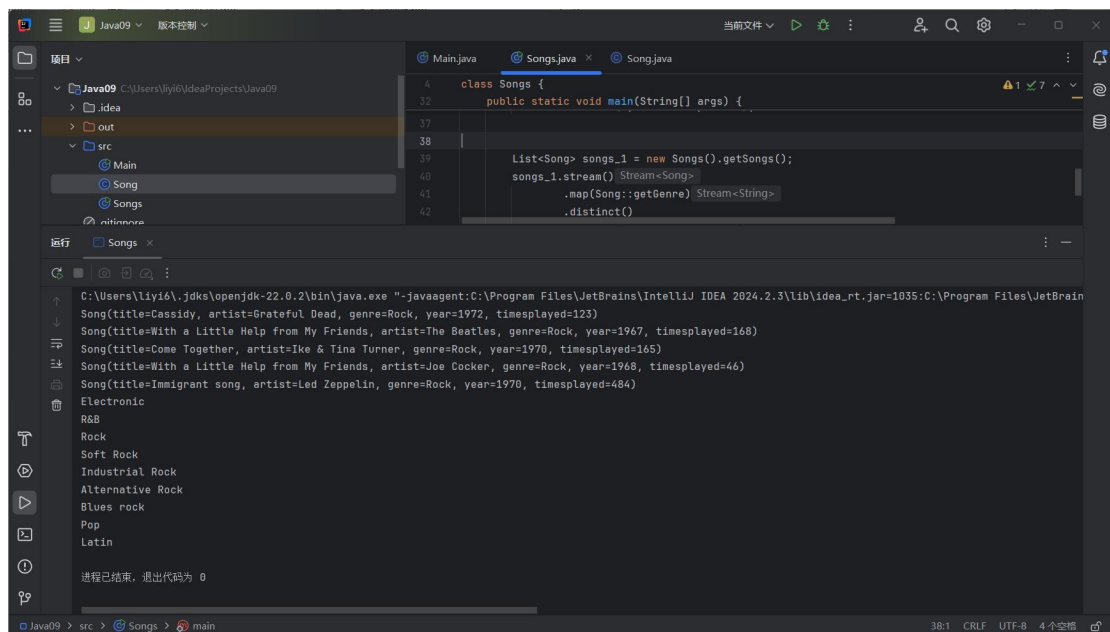
（代码已上传至仓库）

重点如下



1. 使用 `.filter` 操作，调取 `getGenre` 方法（即根据 `genre` 进行筛选），找到所有匹配为 `Rock` 的元素并用 `.forEach` 终端操作进行打印。
2. 使用 `.map` 操作，调取 `getGenre` 方法选取流元素中的 `genre` 参数构成一个新的流，注意，由于“风格 `genre`”这个参数有重复的情况，我额外使用了 `.distinct` 操作来去重。
3. 这里重写了 `toString` 方法，不然又变成了哈希码的结果。

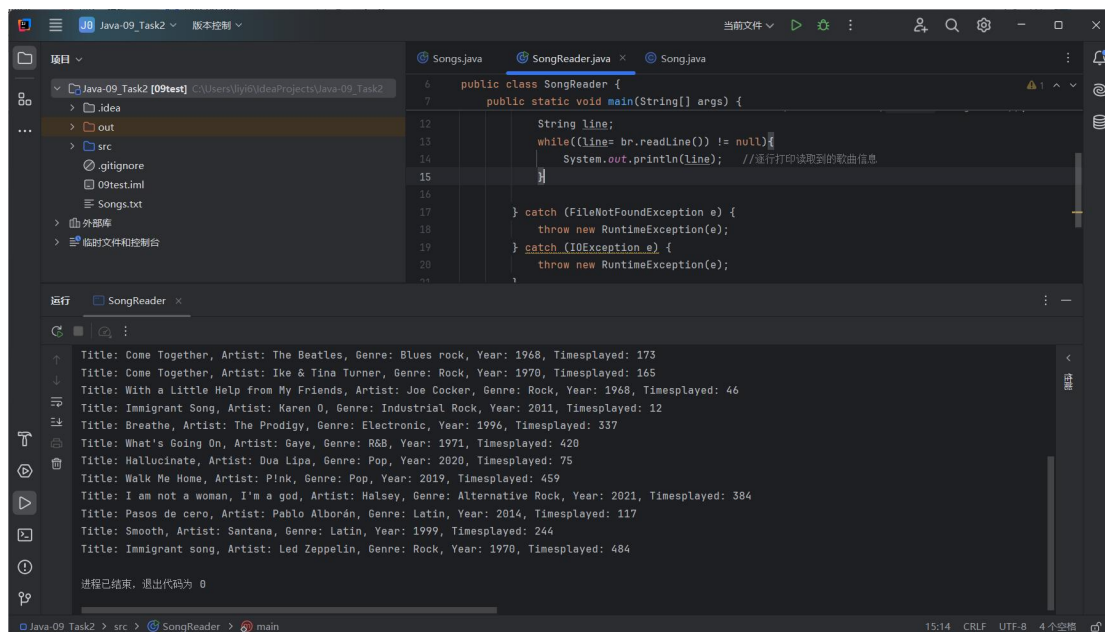
完整的结果如下图：



至于文件串行化保存和读取，做题的时候主要在异常处理方面卡了半天，还是不熟悉，后面查了一堆看了一堆+IDEA 的神辅助才弄好

Task2

(仅包含第二个进阶挑战)
结果如图:



写入后的 txt 文件:



其他的没怎么变，只是用操作流时的最后终端操作使用.collect(Collectors.joining());将流中的

元素都转为字符串，后用 **File Writer** 将已经转换过的字符串写到 **txt** 文件里，当然这里也重写了 **toString** 方法统一了格式，原因和上面一样（又变成类+哈希码）。然后另写了一个 **Song Reader** 程序用 **BufferedReader** 逐行读取文件（简单处理成了纯文本）并打印出来。