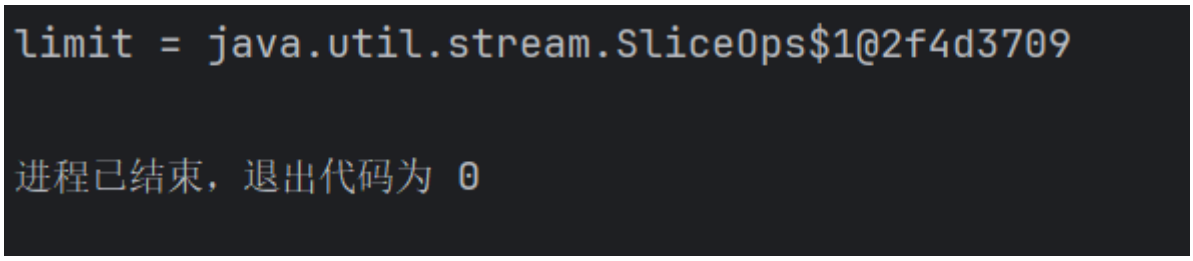
task 1 流API

修改代码

运行截图



可以发现，输出的并不是我们想要的结果，查资料发现直接打印stream对象时，默认调用的是**toString()方法**，返回的是对象的字符串类型信息格式为"**ClassName@hashCode的无符号十六进制表示**"的字符串，以截图为例，java.util.stream.Slice.Ops$1为类名，后面的就是hashcode了，为了得到我们想要的结果，可采用两个办法，

1. 重写toString()方法
2. 用forEach（迭代流中元素） **代码示例**

```
import java.util.List;
import java.util.stream.Stream;
class Main {
    public static void main(String[] args) {
        List<String> strings = List.of("I", "am", "a", "list", "of",
"Strings");/*.of是jdk9以上才有的，我先前的jdk用的还是jdk8，这次换成jdk23了*/
        Stream<String> stream = strings.stream();
        Stream<String> limit = stream.limit(4);
        limit.forEach(System.out::println);/*::用来分隔调用的方法*/
        }
    }
```

输出结果

创建自己的流

1.以集合数组形式来创建流

```
import java.util.ArrayList;
import java.util.List;
import java.util.stream.Stream;
class ain {
```

```
    public static void main(String[] args) {
        List<String> list = new ArrayList<>();
        list.add("时代少年团，我们喜欢你");
        list.add("马嘉祺");
        list.add("丁程鑫");
        list.add("宋亚轩");
        list.stream();/*一般这样就算搞完了，但是我们可以将操作堆叠在一起比如
            .map();
            .sorted();*/
    }
}
```

2.使用java.util.Arrays.stream(T[] array)方法用数组创建流

```
int[] array={1,3,5,6,8};
IntStream stream = Arrays.stream(array);
```

堆叠处理

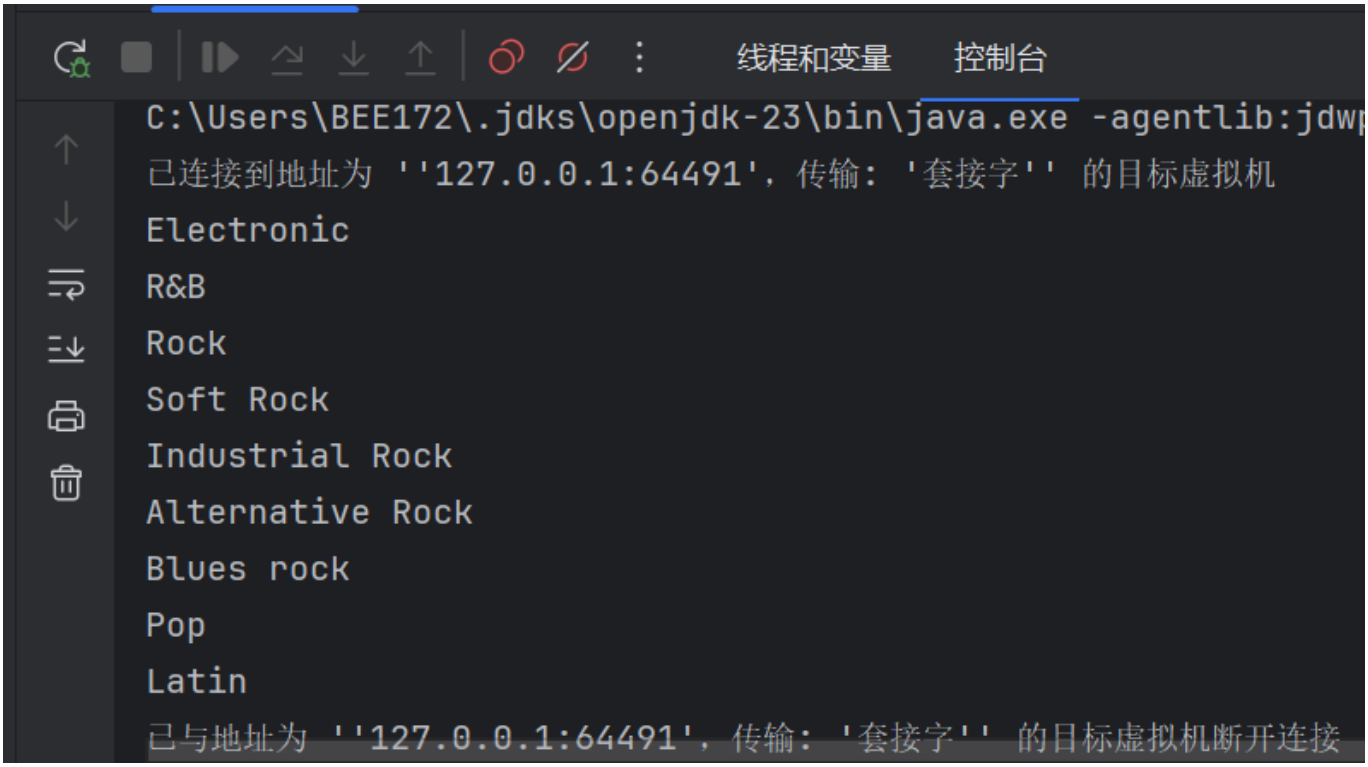
流能够进行堆叠处理与中间操作、终端操作和延迟执行相关，中间操作允许我们调用许多操作但不立即执行，中间操作包含filter，map，sorted，limit，终端操作负责将中间操作一一执行，**值得注意的是**，一经终端使用后，**流就不再使用**，（终端操作如forEach，collect，reduce）。延迟执行指的是只有终端操作被调用后才开始执行流中的操作

LAmbda表达式

LAmbda表达式是对匿名函数的简写形式，可理解为是一段可以传递的代码，这样做更直观了。查资料的LAmbda格式为**实现的这个接口中的抽象方法中的形参列表 -> 抽象方法的处理**，它需要函数式接口（只包含一个抽象方法的接口）的支持，用法和写的格式多样，在题目给的资料网址中都能查到，这里不再赘述。

应用流API

运行截图



有很多方法调用要了解，比如这些 .distinct()， .filter(song -> "Rock".equalsIgnoreCase(song.getGenre()))

（java功能真的太多了，只有我想不到的份😭）

task 2 串行化

Serializable接口

作用非常单一，但也非常重要，这个接口没有任何方法或字段，它只是一个标记，用来标记对象可被串行化。

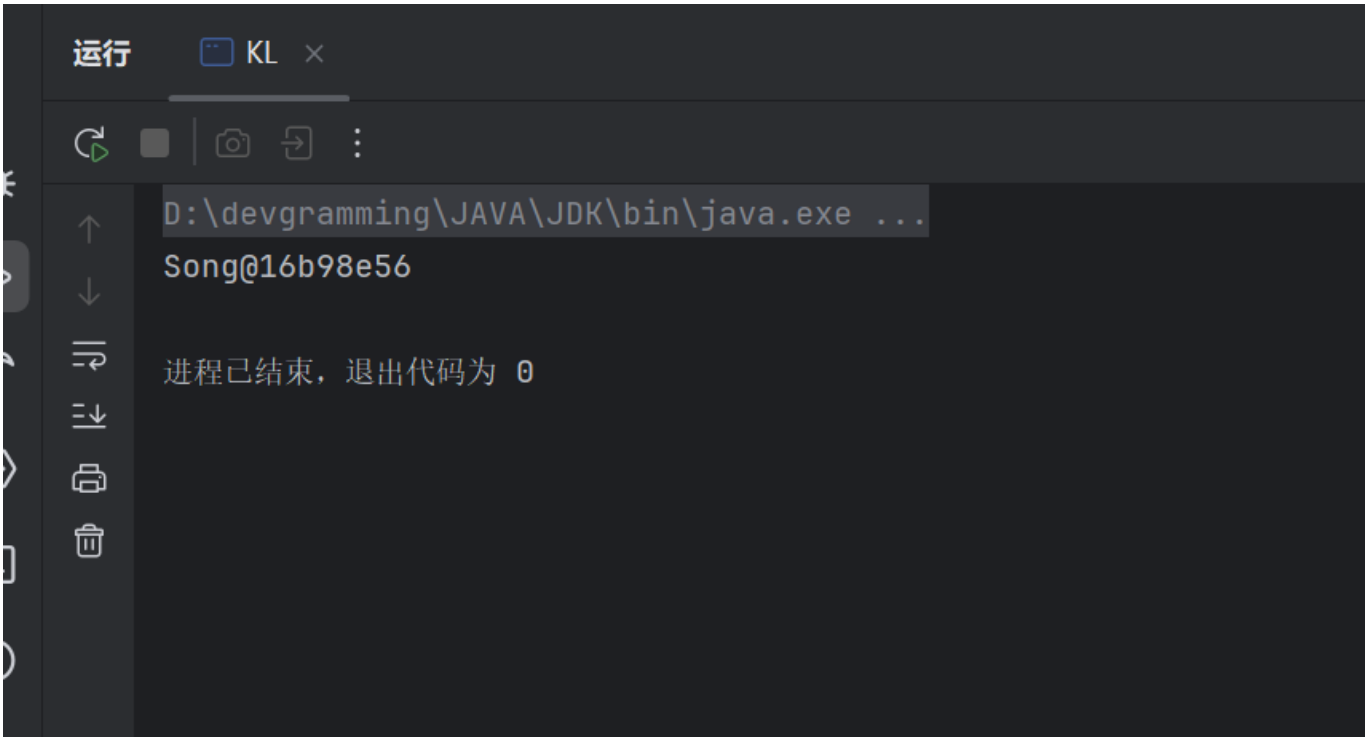
问题来了，什么是串行化？（好像也叫序列化）

串行化

串行化是将对象状态转换为可保持或传输的格式的过程。与串行化相对的是反串行化，它将流转换为对象。这两个过程结合起来，可以轻松地存储和传输数据。 查资料发现使用时最好自己定义一个serialversionUID，否则易抛出InvalidClassException

代码实现

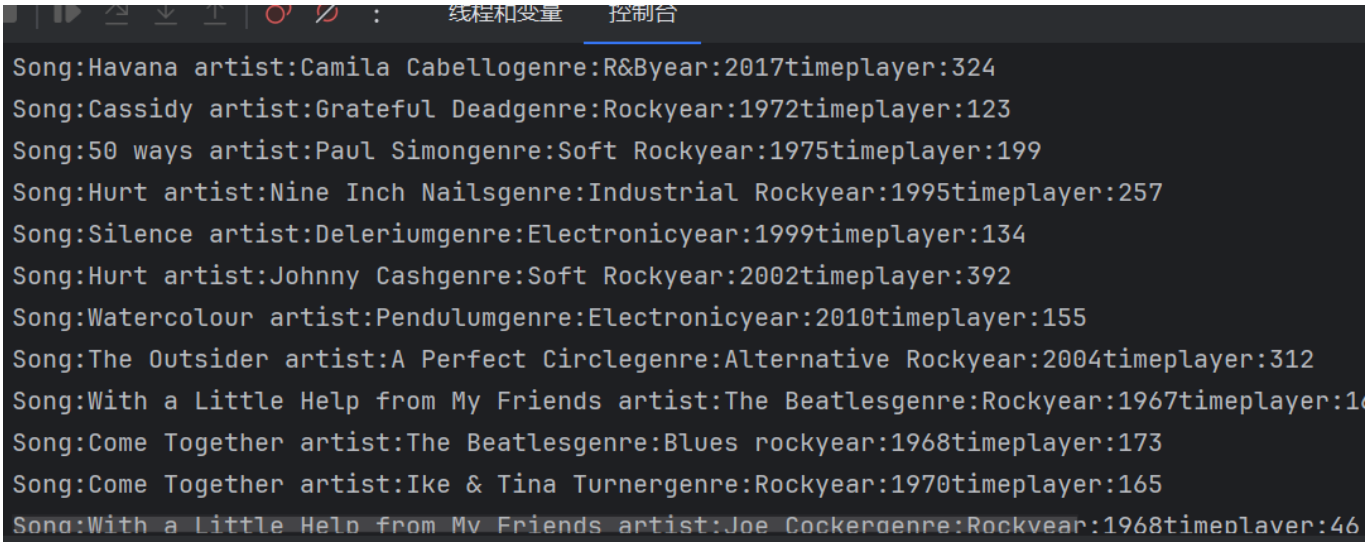
这次就只写一个"歌曲"来进行串行化的操作 运行截图



（成功的返回了对象信息，不过没有重写toSpring()）

进阶挑战——文件I/O

因为FileWriter写入的是String，所以我们写Song类时仍然需要用到Stream 输出结果：发现输出结果是一行的如果要改进的话，可以在读取时不输出，读取完后进行一次for强循环遍历list再进行输出



修改结果

总结

Stream是一个非常有效的操作，可以满足许多要求，（前提是熟练方法的使用），lambda很简洁，但是学习成本很大，我还是不怎么明白使用，读取文件为能写出交互填补了最后的空白